

openCMISS
0.1

Generated by Doxygen 1.5.7.1

Tue Nov 4 15:49:01 2008

Contents

1	openCMISS Documentation	1
1.1	LICENSE	1
2	Obtaining the Code and Setting up the Development Environment	3
2.1	Obtaining the Code and Libraries	4
2.1.1	Obtain the Code	4
2.1.2	Obtain the Libraries	4
2.1.3	Makefile Structure	4
2.2	Programmer documentation	5
2.3	Project Setup	5
2.3.1	On AIX 5.3 (HPC)	5
2.3.1.1	Set environment	5
2.3.1.2	Set MPI	5
2.3.1.3	Compile	5
2.3.2	On Ubuntu 8.04	5
2.3.2.1	Set environment	5
2.3.2.2	Install Compilers	6
2.3.2.3	Compile	6
2.3.3	On Windows XP (Visual Studio 2005)	6
2.3.3.1	Install Compilers	6
2.3.3.2	Install MPI	7
2.3.3.3	Compile and Debug	7
2.3.3.4	Run	7
2.3.4	On Windows Vista (Visual Studio 2008)	7
2.3.4.1	7
2.3.4.2	Compile	7
2.4	Libraries Build (Optional)	8
2.4.1	Compiling PETSc	8

2.4.1.1	Step1: Linux Environment installation and Compiler Environment Set up (For Windows only)	8
2.4.1.2	Step2: Compile PETSC	8
3	Library Commands	11
3.1	OpenCMISS library structure	12
3.2	Top Level - cmis	12
3.3	Basis Routines	12
3.4	Coordinate System Routines	12
3.5	Region Routines	12
3.5.1	Node Routines	12
3.5.2	Mesh Routines	12
3.5.3	Field Routines	12
3.5.4	Equations Set Routines	12
3.6	Problems Routines	12
3.6.1	Solver Routines	12
4	Todo List	13
5	Module Documentation	19
5.1	BASE_ROUTINES::OutputType	19
5.1.1	Detailed Description	19
5.1.2	Variable Documentation	19
5.1.2.1	DIAGNOSTIC_OUTPUT_TYPE	19
5.1.2.2	ERROR_OUTPUT_TYPE	20
5.1.2.3	GENERAL_OUTPUT_TYPE	20
5.1.2.4	HELP_OUTPUT_TYPE	20
5.1.2.5	TIMING_OUTPUT_TYPE	21
5.2	BASE_ROUTINES::FileUnits	22
5.2.1	Detailed Description	23
5.2.2	Variable Documentation	23
5.2.2.1	DIAGNOSTICS_FILE_UNIT	23
5.2.2.2	ECHO_FILE_UNIT	23
5.2.2.3	IO1_FILE_UNIT	23
5.2.2.4	IO2_FILE_UNIT	23
5.2.2.5	IO3_FILE_UNIT	24
5.2.2.6	IO4_FILE_UNIT	24
5.2.2.7	IO5_FILE_UNIT	24

5.2.2.8	LEARN_FILE_UNIT	24
5.2.2.9	OPEN_COMFILE_UNIT	24
5.2.2.10	START_READ_COMFILE_UNIT	25
5.2.2.11	STOP_READ_COMFILE_UNIT	25
5.2.2.12	TEMPORARY_FILE_UNIT	25
5.2.2.13	TIMING_FILE_UNIT	25
5.3	BASE_ROUTINES::DiagnosticTypes	26
5.3.1	Detailed Description	26
5.3.2	Variable Documentation	26
5.3.2.1	ALL_DIAG_TYPE	26
5.3.2.2	FROM_DIAG_TYPE	26
5.3.2.3	IN_DIAG_TYPE	27
5.4	BASE_ROUTINES::TimingTypes	28
5.4.1	Detailed Description	28
5.4.2	Variable Documentation	28
5.4.2.1	ALL_TIMING_TYPE	28
5.4.2.2	FROM_TIMING_TYPE	28
5.4.2.3	IN_TIMING_TYPE	29
5.5	COORDINATE_ROUTINES::CoordinateSystemTypes	30
5.5.1	Detailed Description	30
5.5.2	Variable Documentation	30
5.5.2.1	COORDINATE_CYCLINDRICAL_POLAR_TYPE	30
5.5.2.2	COORDINATE_OBLATE_SPHEROIDAL_TYPE	31
5.5.2.3	COORDINATE_PROLATE_SPHEROIDAL_TYPE	31
5.5.2.4	COORDINATE_RECTANGULAR_CARTESIAN_TYPE	32
5.5.2.5	COORDINATE_SPHERICAL_POLAR_TYPE	32
5.6	COORDINATE_ROUTINES::RadialInterpolations	34
5.6.1	Detailed Description	34
5.6.2	Variable Documentation	34
5.6.2.1	COORDINATE_NO_RADIAL_INTERPOLATION_TYPE	34
5.6.2.2	COORDINATE_RADIAL_CUBED_INTERPOLATION_TYPE	35
5.6.2.3	COORDINATE_RADIAL_INTERPOLATION_TYPE	35
5.6.2.4	COORDINATE_RADIAL_SQUARED_INTERPOLATION_TYPE	35
5.7	COORDINATE_ROUTINES::JacobianType	36
5.7.1	Detailed Description	36
5.7.2	Variable Documentation	36

5.7.2.1	COORDINATE_JACOBIAN_AREA_TYPE	36
5.7.2.2	COORDINATE_JACOBIAN_LINE_TYPE	36
5.7.2.3	COORDINATE_JACOBIAN_VOLUME_TYPE	37
5.8	DOMAIN_MAPPINGS::DomainType	38
5.8.1	Detailed Description	38
5.8.2	Variable Documentation	38
5.8.2.1	DOMAIN_LOCAL_BOUNDARY	38
5.8.2.2	DOMAIN_LOCAL_GHOST	38
5.8.2.3	DOMAIN_LOCAL_INTERNAL	38
5.9	EQUATIONS_SET_CONSTANTS::SetupTypes	40
5.9.1	Detailed Description	41
5.9.2	Variable Documentation	41
5.9.2.1	EQUATIONS_SET_SETUP_ANALYTIC_TYPE	41
5.9.2.2	EQUATIONS_SET_SETUP_DEPENDENT_TYPE	41
5.9.2.3	EQUATIONS_SET_SETUP_EQUATIONS_TYPE	41
5.9.2.4	EQUATIONS_SET_SETUP_FINAL_TYPE	42
5.9.2.5	EQUATIONS_SET_SETUP_FIXED_CONDITIONS_TYPE	42
5.9.2.6	EQUATIONS_SET_SETUP_GEOMETRY_TYPE	42
5.9.2.7	EQUATIONS_SET_SETUP_INITIAL_TYPE	42
5.9.2.8	EQUATIONS_SET_SETUP_MATERIALS_TYPE	43
5.9.2.9	EQUATIONS_SET_SETUP_SOURCE_MATERIALS_TYPE	43
5.9.2.10	EQUATIONS_SET_SETUP_SOURCE_TYPE	43
5.10	EQUATIONS_SET_CONSTANTS::SetupActionTypes	44
5.10.1	Detailed Description	44
5.10.2	Variable Documentation	44
5.10.2.1	EQUATIONS_SET_SETUP_FINISH_ACTION	44
5.10.2.2	EQUATIONS_SET_SETUP_START_ACTION	44
5.11	EQUATIONS_SET_CONSTANTS::FixedConditions	46
5.11.1	Detailed Description	46
5.11.2	Variable Documentation	46
5.11.2.1	EQUATIONS_SET_FIXED_BOUNDARY_CONDITION	46
5.11.2.2	EQUATIONS_SET_MIXED_BOUNDARY_CONDITION	46
5.11.2.3	EQUATIONS_SET_NOT_FIXED	47
5.12	EQUATIONS_SET_CONSTANTS::LinearityTypes	48
5.12.1	Detailed Description	48
5.12.2	Variable Documentation	48

5.12.2.1	EQUATIONS_SET_LINEAR	48
5.12.2.2	EQUATIONS_SET_NONLINEAR	49
5.12.2.3	EQUATIONS_SET_NONLINEAR_BCS	49
5.12.2.4	NUMBER_OF_EQUATIONS_SET_LINEARITY_TYPES	49
5.13	EQUATIONS_SET_CONSTANTS::JacobianCalculationTypes	50
5.13.1	Detailed Description	50
5.13.2	Variable Documentation	50
5.13.2.1	EQUATIONS_SET_JACOBIAN_ANALYTIC_CALCULATED	50
5.13.2.2	EQUATIONS_SET_JACOBIAN_FD_CALCULATED	50
5.13.2.3	EQUATIONS_SET_JACOBIAN_NOT_CALCULATED	51
5.14	EQUATIONS_SET_CONSTANTS::TimeDependenceTypes	52
5.14.1	Detailed Description	52
5.14.2	Variable Documentation	52
5.14.2.1	EQUATIONS_SET_DYNAMIC	52
5.14.2.2	EQUATIONS_SET_QUASISTATIC	52
5.14.2.3	EQUATIONS_SET_STATIC	53
5.14.2.4	NUMBER_OF_EQUATIONS_SET_TIME_TYPES	53
5.15	EQUATIONS_SET_CONSTANTS::SolutionMethods	54
5.15.1	Detailed Description	54
5.15.2	Variable Documentation	54
5.15.2.1	EQUATIONS_SET_BEM SOLUTION_METHOD	54
5.15.2.2	EQUATIONS_SET_FD SOLUTION_METHOD	55
5.15.2.3	EQUATIONS_SET_FEM SOLUTION_METHOD	55
5.15.2.4	EQUATIONS_SET_FV SOLUTION_METHOD	55
5.15.2.5	EQUATIONS_SET_GFEM SOLUTION_METHOD	56
5.15.2.6	EQUATIONS_SET_GFV SOLUTION_METHOD	56
5.15.2.7	NUMBER_OF_EQUATIONS_SET_SOLUTION_METHODS	56
5.16	EQUATIONS_SET_CONSTANTS::OutputTypes	57
5.16.1	Detailed Description	57
5.16.2	Variable Documentation	57
5.16.2.1	EQUATIONS_SET_ELEMENT_MATRIX_OUTPUT	57
5.16.2.2	EQUATIONS_SET_MATRIX_OUTPUT	57
5.16.2.3	EQUATIONS_SET_NO_OUTPUT	58
5.16.2.4	EQUATIONS_SET_TIMING_OUTPUT	58
5.17	EQUATIONS_SET_CONSTANTS::SparsityTypes	59
5.17.1	Detailed Description	59

5.17.2	Variable Documentation	59
5.17.2.1	EQUATIONS_SET_FULL_MATRICES	59
5.17.2.2	EQUATIONS_SET_SPARSE_MATRICES	59
5.18	FIELD_ROUTINES::DependentTypes	60
5.18.1	Detailed Description	61
5.18.2	Function Documentation	62
5.18.2.1	LAPLACE_EQUATION_ANALYTIC_CALCULATE	62
5.18.2.2	LAPLACE_EQUATION_ANALYTIC_PARAMETER_SET_UPDATE	62
5.18.2.3	LAPLACE_EQUATION_EQUATIONS_SET_SETUP	63
5.18.2.4	LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP	63
5.18.2.5	LAPLACE_EQUATION_EQUATIONS_SET_SUBTYPE_SET	64
5.18.2.6	LAPLACE_EQUATIONFINITE_ELEMENT_CALCULATE	65
5.18.2.7	LAPLACE_EQUATION_FIXED_CONDITIONS_	66
5.18.2.8	LAPLACE_EQUATION_INTE	66
5.18.2.9	LAPLACE_EQUATION_PROBLEM_SETUP	67
5.18.2.10	LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP	67
5.18.2.11	LAPLACE_EQUATION_PROBLEM_SUBTYPE_SET	68
5.18.3	Variable Documentation	69
5.18.3.1	FIELD_DEPENDENT_TYPE	69
5.18.3.2	FIELD_INDEPENDENT_TYPE	69
5.18.3.3	LAPLACE_EQUATION_THREE_DIM_1	69
5.18.3.4	LAPLACE_EQUATION_THREE_DIM_2	69
5.18.3.5	LAPLACE_EQUATION_TWO_DIM_1	69
5.18.3.6	LAPLACE_EQUATION_TWO_DIM_2	70
5.19	FIELD_ROUTINES::DimensionTypes	71
5.19.1	Detailed Description	71
5.19.2	Variable Documentation	71
5.19.2.1	FIELD_SCALAR_DIMENSION_TYPE	71
5.19.2.2	FIELD_TENSOR_DIMENSION_TYPE	71
5.19.2.3	FIELD_VECTOR_DIMENSION_TYPE	72
5.20	FIELD_ROUTINES::FieldTypes	73
5.20.1	Detailed Description	73
5.20.2	Variable Documentation	73
5.20.2.1	FIELD_FIBRE_TYPE	73
5.20.2.2	FIELD_GENERAL_TYPE	73
5.20.2.3	FIELD_GEOMETRIC_TYPE	74

5.20.2.4 FIELD_MATERIAL_TYPE	74
5.21 FIELD_ROUTINES::InterpolationTypes	75
5.21.1 Detailed Description	75
5.21.2 Variable Documentation	75
5.21.2.1 FIELD_CONSTANT_INTERPOLATION	75
5.21.2.2 FIELD_ELEMENT_BASED_INTERPOLATION	76
5.21.2.3 FIELD_GAUSS_POINT_BASED_INTERPOLATION	76
5.21.2.4 FIELD_GRID_POINT_BASED_INTERPOLATION	76
5.21.2.5 FIELD_NODE_BASED_INTERPOLATION	76
5.22 FIELD_ROUTINES::VariableTypes	78
5.22.1 Detailed Description	78
5.22.2 Variable Documentation	78
5.22.2.1 FIELD_NORMAL_VARIABLE_TYPE	78
5.22.2.2 FIELD_NUMBER_OF_VARIABLE_TYPES	79
5.22.2.3 FIELD_STANDARD_VARIABLE_TYPE	79
5.22.2.4 FIELD_TIME_DERIV1_VARIABLE_TYPE	79
5.22.2.5 FIELD_TIME_DERIV2_VARIABLE_TYPE	80
5.23 FIELD_ROUTINES::DofTypes	81
5.23.1 Detailed Description	81
5.23.2 Variable Documentation	81
5.23.2.1 FIELD_CONSTANT_DOF_TYPE	81
5.23.2.2 FIELD_ELEMENT_DOF_TYPE	81
5.23.2.3 FIELD_NODE_DOF_TYPE	82
5.23.2.4 FIELD_POINT_DOF_TYPE	82
5.24 FIELD_ROUTINES::ParameterSetTypes	83
5.24.1 Detailed Description	83
5.24.2 Variable Documentation	83
5.24.2.1 FIELD_ANALYTIC_SET_TYPE	83
5.24.2.2 FIELD_BOUNDARY_CONDITIONS_SET_TYPE	84
5.24.2.3 FIELD_INITIAL_CONDITIONS_SET_TYPE	84
5.24.2.4 FIELD_NUMBER_OF_SET_TYPES	84
5.24.2.5 FIELD_VALUES_SET_TYPE	84
5.25 FIELD_ROUTINES::ScalingTypes	86
5.25.1 Detailed Description	86
5.25.2 Variable Documentation	86
5.25.2.1 FIELD_ARC_LENGTH_SCALING	86

5.25.2.2	<code>FIELD_ARITHMETIC_MEAN_SCALING</code>	87
5.25.2.3	<code>FIELD_HARMONIC_MEAN_SCALING</code>	87
5.25.2.4	<code>FIELD_NO_SCALING</code>	87
5.25.2.5	<code>FIELD_UNIT_SCALING</code>	87
5.26	<code>GENERATED_MESH_ROUTINES::GeneratedMeshTypes</code>	88
5.26.1	Detailed Description	88
5.26.2	Variable Documentation	88
5.26.2.1	<code>GENERATED_MESH_FRACTAL_TREE_MESH_TYPE</code>	88
5.26.2.2	<code>GENERATED_MESH_POLAR_MESH_TYPE</code>	88
5.26.2.3	<code>GENERATED_MESH_REGULAR_MESH_TYPE</code>	89
5.27	<code>INPUT_OUTPUT::MatrixNameIndexFormat</code>	90
5.27.1	Detailed Description	90
5.27.2	Variable Documentation	90
5.27.2.1	<code>WRITE_STRING_MATRIX_NAME_AND_INDICES</code>	90
5.27.2.2	<code>WRITE_STRING_MATRIX_NAME_ONLY</code>	90
5.28	<code>KINDS::IntegerKinds</code>	92
5.28.1	Detailed Description	92
5.28.2	Variable Documentation	92
5.28.2.1	<code>INTG</code>	92
5.28.2.2	<code>LINTG</code>	92
5.28.2.3	<code>PTR</code>	93
5.28.2.4	<code>SINTG</code>	94
5.29	<code>KINDS::RealKinds</code>	95
5.29.1	Detailed Description	95
5.29.2	Variable Documentation	95
5.29.2.1	<code>DP</code>	95
5.29.2.2	<code>QP</code>	95
5.29.2.3	<code>SP</code>	96
5.30	<code>KINDS::ComplexKinds</code>	97
5.30.1	Detailed Description	97
5.30.2	Variable Documentation	97
5.30.2.1	<code>_DP</code>	97
5.30.2.2	<code>_SP</code>	98
5.30.2.3	<code>DPC</code>	99
5.30.2.4	<code>SPC</code>	99
5.31	<code>LISTS::DataType</code>	100

5.31.1	Detailed Description	100
5.31.2	Variable Documentation	100
5.31.2.1	LIST_DP_TYPE	100
5.31.2.2	LIST_INTG_TYPE	100
5.31.2.3	LIST_SP_TYPE	101
5.32	LISTS::SortingOrder	102
5.32.1	Detailed Description	102
5.32.2	Variable Documentation	102
5.32.2.1	LIST_SORT_ASCENDING_TYPE	102
5.32.2.2	LIST_SORT_DESCENDING_TYPE	102
5.32.2.3	LIST_UNSORTED_TYPE	103
5.33	LISTS::SortingMethod	104
5.33.1	Detailed Description	104
5.33.2	Variable Documentation	104
5.33.2.1	LIST_BUBBLE_SORT_METHOD	104
5.33.2.2	LIST_HEAP_SORT_METHOD	104
5.33.2.3	LIST_SHELL_SORT_METHOD	105
5.34	MATRIX_VECTOR::DataTypes	106
5.34.1	Detailed Description	106
5.34.2	Variable Documentation	106
5.34.2.1	MATRIX_VECTOR_DP_TYPE	106
5.34.2.2	MATRIX_VECTOR_INTG_TYPE	107
5.34.2.3	MATRIX_VECTOR_L_TYPE	107
5.34.2.4	MATRIX_VECTOR_SP_TYPE	108
5.35	MATRIX_VECTOR::StorageTypes	109
5.35.1	Detailed Description	109
5.35.2	Variable Documentation	109
5.35.2.1	MATRIX_BLOCK_STORAGE_TYPE	109
5.35.2.2	MATRIX_COLUMN_MAJOR_STORAGE_TYPE	110
5.35.2.3	MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE	110
5.35.2.4	MATRIX_COMPRESSED_ROW_STORAGE_TYPE	111
5.35.2.5	MATRIX_DIAGONAL_STORAGE_TYPE	111
5.35.2.6	MATRIX_ROW_COLUMN_STORAGE_TYPE	111
5.35.2.7	MATRIX_ROW_MAJOR_STORAGE_TYPE	112
5.36	MESH_ROUTINES::DecompositionTypes	113
5.36.1	Detailed Description	113

5.36.2	Variable Documentation	113
5.36.2.1	DECOMPOSITION_ALL_TYPE	113
5.36.2.2	DECOMPOSITION_CALCULATED_TYPE	113
5.36.2.3	DECOMPOSITION_USER_DEFINED_TYPE	114
5.37	PROBLEM_CONSTANTS::SetupTypes	115
5.37.1	Detailed Description	115
5.37.2	Variable Documentation	115
5.37.2.1	PROBLEM_SETUP_CONTROL_TYPE	115
5.37.2.2	PROBLEM_SETUP_INITIAL_TYPE	116
5.37.2.3	PROBLEM_SETUP SOLUTION_TYPE	116
5.37.2.4	PROBLEM_SETUP_SOLVER_TYPE	116
5.38	PROBLEM_CONSTANTS::SetupActionTypes	117
5.38.1	Detailed Description	117
5.38.2	Variable Documentation	117
5.38.2.1	PROBLEM_SETUP_DO_ACTION	117
5.38.2.2	PROBLEM_SETUP_FINISH_ACTION	117
5.38.2.3	PROBLEM_SETUP_START_ACTION	118
5.39	PROBLEM_CONSTANTS::SolutionOutputTypes	119
5.39.1	Detailed Description	119
5.39.2	Variable Documentation	119
5.39.2.1	PROBLEM SOLUTION MATRIX_OUTPUT	119
5.39.2.2	PROBLEM SOLUTION NO_OUTPUT	119
5.39.2.3	PROBLEM SOLUTION TIMING_OUTPUT	120
5.40	PROBLEM_CONSTANTS::SolutionLinearityTypes	121
5.40.1	Detailed Description	121
5.40.2	Variable Documentation	121
5.40.2.1	PROBLEM SOLUTION LINEAR	121
5.40.2.2	PROBLEM SOLUTION NONLINEAR	121
5.41	PROBLEM_CONSTANTS::SolverOutputTypes	123
5.41.1	Detailed Description	123
5.41.2	Variable Documentation	123
5.41.2.1	PROBLEM SOLVER NO_OUTPUT	123
5.41.2.2	PROBLEM SOLVER SOLVER_OUTPUT	123
5.41.2.3	PROBLEM SOLVER TIMING_OUTPUT	124
5.42	PROBLEM_CONSTANTS::SolverSparsityTypes	125
5.42.1	Detailed Description	125

5.42.2	Variable Documentation	125
5.42.2.1	PROBLEM_SOLVER_FULL_MATRICES	125
5.42.2.2	PROBLEM_SOLVER_SPARSE_MATRICES	125
5.43	PROBLEM_ROUTINES::LinearityTypes	126
5.43.1	Detailed Description	126
5.43.2	Variable Documentation	126
5.43.2.1	NUMBER_OF_PROBLEM_LINEARITIES	126
5.43.2.2	PROBLEM_LINEAR	126
5.43.2.3	PROBLEM_NONLINEAR	127
5.43.2.4	PROBLEM_NONLINEAR_BCS	127
5.44	PROBLEM_ROUTINES::TimeDepedenceTypes	128
5.44.1	Detailed Description	128
5.44.2	Variable Documentation	128
5.44.2.1	NUMBER_OF_PROBLEM_TIME_TYPES	128
5.44.2.2	PROBLEM_DYNAMIC	128
5.44.2.3	PROBLEM_QUASISTATIC	129
5.44.2.4	PROBLEM_STATIC	129
5.45	SOLVER_ROUTINES::SolverTypes	130
5.45.1	Detailed Description	130
5.45.2	Variable Documentation	130
5.45.2.1	SOLVER_EIGENPROBLEM_TYPE	130
5.45.2.2	SOLVER_LINEAR_TYPE	130
5.45.2.3	SOLVER_NONLINEAR_TYPE	131
5.45.2.4	SOLVER_TIME_INTEGRATION_TYPE	131
5.46	SOLVER_ROUTINES::SolverLibraries	132
5.46.1	Detailed Description	132
5.46.2	Variable Documentation	132
5.46.2.1	SOLVER_CMISS_LIBRARY	132
5.46.2.2	SOLVER_PETSC_LIBRARY	132
5.47	SOLVER_ROUTINES::LinearSolverTypes	134
5.47.1	Detailed Description	134
5.47.2	Variable Documentation	134
5.47.2.1	SOLVER_LINEAR_DIRECT_SOLVE_TYPE	134
5.47.2.2	SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE	134
5.48	SOLVER_ROUTINES::DirectLinearSolverTypes	136
5.48.1	Detailed Description	136

5.48.2	Variable Documentation	136
5.48.2.1	SOLVER_DIRECT_CHOLESKY	136
5.48.2.2	SOLVER_DIRECT_LU	136
5.48.2.3	SOLVER_DIRECT_SVD	137
5.49	SOLVER_ROUTINES::IterativeLinearSolverTypes	138
5.49.1	Detailed Description	138
5.49.2	Variable Documentation	138
5.49.2.1	SOLVER_ITERATIVE_BICGSTAB	138
5.49.2.2	SOLVER_ITERATIVE_BICONJUGATE_GRADIENT	139
5.49.2.3	SOLVER_ITERATIVE_CHEBYCHEV	139
5.49.2.4	SOLVER_ITERATIVE_CONJGRAD_SQUARED	139
5.49.2.5	SOLVER_ITERATIVE_CONJUGATE_GRADIENT	139
5.49.2.6	SOLVER_ITERATIVE_GMRES	140
5.49.2.7	SOLVER_ITERATIVE_RICHARDSON	140
5.50	SOLVER_ROUTINES::IterativePreconditionerTypes	141
5.50.1	Detailed Description	141
5.50.2	Variable Documentation	141
5.50.2.1	SOLVER_ITERATIVE_ADDITIVE_SCHWARZ_PRECONDITIONER	141
5.50.2.2	SOLVER_ITERATIVE_BLOCK_JACOBI_PRECONDITIONER . . .	142
5.50.2.3	SOLVER_ITERATIVE_INCOMPLETE_CHOLESKY_- PRECONDITIONER	142
5.50.2.4	SOLVER_ITERATIVE_INCOMPLETE_LU_PRECONDITIONER . .	142
5.50.2.5	SOLVER_ITERATIVE_JACOBI_PRECONDITIONER	143
5.50.2.6	SOLVER_ITERATIVE_NO_PRECONDITIONER	143
5.50.2.7	SOLVER_ITERATIVE_SOR_PRECONDITIONER	143
5.51	SOLVER_ROUTINES::NonlinearSolverTypes	144
5.51.1	Detailed Description	144
5.51.2	Variable Documentation	144
5.51.2.1	SOLVER_NONLINEAR_LINESearch	144
5.51.2.2	SOLVER_NONLINEAR_TRUSTREGION	144
5.52	SOLVER_ROUTINES::NonlinearLineSearchTypes	146
5.52.1	Detailed Description	146
5.52.2	Variable Documentation	146
5.52.2.1	SOLVER_NONLINEAR_CUBIC_LINESearch	146
5.52.2.2	SOLVER_NONLINEAR_NO_LINESearch	147
5.52.2.3	SOLVER_NONLINEAR_NONORMS_LINESearch	147
5.52.2.4	SOLVER_NONLINEAR_QUADRATIC_LINESearch	147

5.53	SOLVER_ROUTINES::JacobianCalculationTypes	148
5.53.1	Detailed Description	148
5.53.2	Variable Documentation	148
5.53.2.1	SOLVER_NONLINEAR_JACOBIAN_ANALYTIC_CALCULATED .	148
5.53.2.2	SOLVER_NONLINEAR_JACOBIAN_FD_CALCULATED	148
5.53.2.3	SOLVER_NONLINEAR_JACOBIAN_NOT_CALCULATED	149
5.54	SOLVER_ROUTINES::OutputTypes	150
5.54.1	Detailed Description	150
5.54.2	Variable Documentation	150
5.54.2.1	SOLVER_MATRIX_OUTPUT	150
5.54.2.2	SOLVER_NO_OUTPUT	150
5.54.2.3	SOLVER_SOLVER_OUTPUT	151
5.54.2.4	SOLVER_TIMING_OUTPUT	151
5.55	SOLVER_ROUTINES::SparsityTypes	152
5.55.1	Detailed Description	152
5.55.2	Variable Documentation	152
5.55.2.1	SOLVER_FULL_MATRICES	152
5.55.2.2	SOLVER_SPARSE_MATRICES	152
5.56	TREES::TreeNodeColourTypes	154
5.56.1	Detailed Description	154
5.56.2	Variable Documentation	154
5.56.2.1	TREE_BLACK_NODE	154
5.56.2.2	TREE_RED_NODE	154
5.57	TREES::TreeNodeInsertStatus	155
5.57.1	Detailed Description	155
5.57.2	Variable Documentation	155
5.57.2.1	TREE_NODE_DUPLICATE_KEY	155
5.57.2.2	TREE_NODE_INSERT_SUCESSFUL	155
5.58	TREES::TreeInsertTypes	156
5.58.1	Detailed Description	156
5.58.2	Variable Documentation	156
5.58.2.1	TREE_DUPLICATES_ALLOWED_TYPE	156
5.58.2.2	TREE_NO_DUPLICATES_ALLOWED	156
6	Namespace Documentation	157
6.1	ANALYTIC_ANALYSIS_ROUTINES Namespace Reference	157
6.1.1	Detailed Description	158

6.1.2	Function Documentation	159
6.1.2.1	ANALYTIC_ANALYSIS_CALCULATE	159
6.1.2.2	ANALYTIC_ANALYSIS_EXPORT	159
6.1.2.3	ANALYTIC_ANALYSIS_FORTRAN_FILE_OPEN	159
6.1.2.4	ANALYTIC_ANALYSIS_FORTRAN_FILE_WRITE_STRING	160
6.1.2.5	ANALYTIC_ANALYSIS_INTEGRAL_ABSOLUTE_ERROR_GET	160
6.1.2.6	ANALYTIC_ANALYSIS_INTEGRAL_ANALYTIC_VALUE_GET	160
6.1.2.7	ANALYTIC_ANALYSIS_INTEGRAL_NUMERICAL_VALUE_GET	161
6.1.2.8	ANALYTIC_ANALYSIS_INTEGRAL_PERCENT_ERROR_GET	161
6.1.2.9	ANALYTIC_ANALYSIS_INTEGRAL_RELATIVE_ERROR_GET	162
6.1.2.10	ANALYTIC_ANALYSIS_NODE_ABSOLUTE_ERROR_GET	162
6.1.2.11	ANALYTIC_ANALYSIS_NODE_ANALYTIC_VALUE_GET	162
6.1.2.12	ANALYTIC_ANALYSIS_NODE_NUMERICAL_VALUE_GET	163
6.1.2.13	ANALYTIC_ANALYSIS_NODE_PERCENT_ERROR_GET	163
6.1.2.14	ANALYTIC_ANALYSIS_NODE_RELATIVE_ERROR_GET	164
6.1.2.15	ANALYTIC_ANALYSIS_RMS_ABSOLUTE_ERROR_GET	164
6.1.2.16	ANALYTIC_ANALYSIS_RMS_PERCENT_ERROR_GET	165
6.1.2.17	ANALYTIC_ANALYSIS_RMS_RELATIVE_ERROR_GET	165
6.2	BASE_ROUTINES Namespace Reference	166
6.2.1	Detailed Description	170
6.2.2	Function Documentation	170
6.2.2.1	BASE_ROUTINES_FINALISE	170
6.2.2.2	BASE_ROUTINES_INITIALISE	171
6.2.2.3	DIAGNOSTICS_SET_OFF	171
6.2.2.4	DIAGNOSTICS_SET_ON	171
6.2.2.5	ENTERS	172
6.2.2.6	ERRORS	182
6.2.2.7	EXITS	194
6.2.2.8	FLAG_ERROR_C	204
6.2.2.9	FLAG_ERROR_VS	204
6.2.2.10	FLAG_WARNING_C	205
6.2.2.11	FLAG_WARNING_VS	205
6.2.2.12	OUTPUT_SET_OFF	205
6.2.2.13	OUTPUT_SET_ON	205
6.2.2.14	TIMING_SET_OFF	206
6.2.2.15	TIMING_SET_ON	206

6.2.2.16	TIMING_SUMMARY_OUTPUT	206
6.2.2.17	WRITE_STR	207
6.2.3	Variable Documentation	208
6.2.3.1	DIAG_ALL_SUBROUTINES	208
6.2.3.2	DIAG_FILE_OPEN	208
6.2.3.3	DIAG_FROM_SUBROUTINE	208
6.2.3.4	DIAG_OR_TIMING	208
6.2.3.5	DIAG_ROUTINE_LIST	208
6.2.3.6	DIAGNOSTICS	208
6.2.3.7	DIAGNOSTICS1	208
6.2.3.8	DIAGNOSTICS2	209
6.2.3.9	DIAGNOSTICS3	209
6.2.3.10	DIAGNOSTICS4	209
6.2.3.11	DIAGNOSTICS5	209
6.2.3.12	DIAGNOSTICS_LEVEL1	209
6.2.3.13	DIAGNOSTICS_LEVEL2	210
6.2.3.14	DIAGNOSTICS_LEVEL3	210
6.2.3.15	DIAGNOSTICS_LEVEL4	210
6.2.3.16	DIAGNOSTICS_LEVEL5	210
6.2.3.17	ECHO_OUTPUT	210
6.2.3.18	MAX_OUTPUT_LINES	210
6.2.3.19	OP_STRING	210
6.2.3.20	ROUTINE_STACK	211
6.2.3.21	TIMING	211
6.2.3.22	TIMING_ALL_SUBROUTINES	211
6.2.3.23	TIMING_FILE_OPEN	212
6.2.3.24	TIMING_FROM_SUBROUTINE	212
6.2.3.25	TIMING_ROUTINE_LIST	212
6.2.3.26	TIMING_SUMMARY	212
6.3	BINARY_FILE Namespace Reference	213
6.3.1	Detailed Description	214
6.3.2	Function Documentation	214
6.3.2.1	CLOSE_BINARY_FILE	214
6.3.2.2	CLOSE_CMISS_BINARY_FILE	215
6.3.2.3	INQUIRE_EOF_BINARY_FILE	215
6.3.2.4	INQUIRE_OPEN_BINARY_FILE	215

6.3.2.5	OPEN_BINARY_FILE	215
6.3.2.6	OPEN_CMISS_BINARY_FILE	215
6.3.2.7	READ_BINARY_FILE_CHARACTER	216
6.3.2.8	READ_BINARY_FILE_DP	216
6.3.2.9	READ_BINARY_FILE_DP1	216
6.3.2.10	READ_BINARY_FILE_DPC	216
6.3.2.11	READ_BINARY_FILE_DPC1	217
6.3.2.12	READ_BINARY_FILE_INTG	217
6.3.2.13	READ_BINARY_FILE_INTG1	217
6.3.2.14	READ_BINARY_FILE_LINTG	217
6.3.2.15	READ_BINARY_FILE_LINTG1	218
6.3.2.16	READ_BINARY_FILE_LOGICAL	218
6.3.2.17	READ_BINARY_FILE_LOGICAL1	218
6.3.2.18	READ_BINARY_FILE_SINTG	218
6.3.2.19	READ_BINARY_FILE_SINTG1	218
6.3.2.20	READ_BINARY_FILE_SP	219
6.3.2.21	READ_BINARY_FILE_SP1	219
6.3.2.22	READ_BINARY_FILE_SPC	219
6.3.2.23	READ_BINARY_FILE_SPC1	219
6.3.2.24	READ_BINARY_TAG_HEADER	220
6.3.2.25	RESET_BINARY_NUMBER_TAGS	220
6.3.2.26	SET_BINARY_FILE	220
6.3.2.27	SKIP_BINARY_FILE	220
6.3.2.28	SKIP_BINARY_TAGS	220
6.3.2.29	SKIP_CM_BINARY_HEADER	221
6.3.2.30	WRITE_BINARY_FILE_CHARACTER	221
6.3.2.31	WRITE_BINARY_FILE_DP	221
6.3.2.32	WRITE_BINARY_FILE_DP1	221
6.3.2.33	WRITE_BINARY_FILE_DPC	221
6.3.2.34	WRITE_BINARY_FILE_DPC1	222
6.3.2.35	WRITE_BINARY_FILE_INTG	222
6.3.2.36	WRITE_BINARY_FILE_INTG1	222
6.3.2.37	WRITE_BINARY_FILE_LINTG	222
6.3.2.38	WRITE_BINARY_FILE_LINTG1	223
6.3.2.39	WRITE_BINARY_FILE_LOGICAL	223
6.3.2.40	WRITE_BINARY_FILE_LOGICAL1	223

6.3.2.41	WRITE_BINARY_FILE_SINTG	223
6.3.2.42	WRITE_BINARY_FILE_SINTG1	223
6.3.2.43	WRITE_BINARY_FILE_SP	224
6.3.2.44	WRITE_BINARY_FILE_SP1	224
6.3.2.45	WRITE_BINARY_FILE_SPC	224
6.3.2.46	WRITE_BINARY_FILE_SPC1	224
6.3.2.47	WRITE_BINARY_TAG_HEADER	225
6.3.3	Variable Documentation	225
6.3.3.1	BINARY_FILE_READABLE	225
6.3.3.2	BINARY_FILE_USED	225
6.3.3.3	BINARY_FILE_WRITABLE	225
6.3.3.4	CMISS_BINARY_FILE_HEADER	225
6.3.3.5	CMISS_BINARY_HISTORY_FILE	225
6.3.3.6	CMISS_BINARY_IDENTITY	225
6.3.3.7	CMISS_BINARY_IDENTITY_HEADER	225
6.3.3.8	CMISS_BINARY_MACHINE_HEADER	225
6.3.3.9	CMISS_BINARY_MATRIX_FILE	226
6.3.3.10	CMISS_BINARY_SIGNAL_FILE	226
6.3.3.11	FILE_BEGINNING	226
6.3.3.12	FILE_CHANGE_ENDIAN	226
6.3.3.13	FILE_CURRENT	226
6.3.3.14	FILE_END	226
6.3.3.15	FILE SAME_ENDIAN	226
6.3.3.16	MAX_NUM_BINARY_FILES	226
6.4	BLAS Namespace Reference	227
6.4.1	Detailed Description	227
6.5	CLASSICAL_FIELD_ROUTINES Namespace Reference	228
6.5.1	Detailed Description	228
6.5.2	Function Documentation	229
6.5.2.1	CL	229
6.5.2.2	CL	229
6.5.2.3	CLASSICAL_FIELD_EQUATIONS_SET_SETUP	230
6.5.2.4	CLASSICAL_FIELDFINITE_ELEMENT_CALCULATE	230
6.5.2.5	CLASSICAL_FIELDFINITE_ELEMENT_JACOBIAN_EVALUATE .	231
6.5.2.6	CLASSICAL_FIELDFINITE_ELEMENT_RESIDUAL_EVALUATE .	231
6.5.2.7	CLASSICAL_FIELDPROBLEM_CLASS_TYPE_GET	232

6.5.2.8	CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_SET	232
6.5.2.9	CLASSICAL_FIELD_PROBLEM_SETUP	233
6.6	CMISS Namespace Reference	234
6.6.1	Detailed Description	234
6.6.2	Function Documentation	234
6.6.2.1	CMISS_FINALISE	234
6.6.2.2	CMISS_INITIALISE	235
6.6.2.3	CMISS_WRITE_ERROR	235
6.6.3	Variable Documentation	235
6.6.3.1	CMISS_BUILD_VERSION	235
6.6.3.2	CMISS_MAJOR_VERSION	235
6.6.3.3	CMISS_MINOR_VERSION	235
6.6.3.4	CMISS_REVISION_VERSION	235
6.7	CMISS_MPI Namespace Reference	236
6.7.1	Detailed Description	236
6.7.2	Function Documentation	236
6.7.2.1	MPI_ERROR_CHECK	236
6.8	CMISS_PARMETIS Namespace Reference	237
6.8.1	Detailed Description	237
6.8.2	Function Documentation	237
6.8.2.1	PARMETIS_PARTKWAY	237
6.8.2.2	PARMETIS_PARTMESHKWAY	238
6.9	CMISS_PETSC Namespace Reference	239
6.9.1	Detailed Description	246
6.9.2	Function Documentation	246
6.9.2.1	PETSC_ERRORHANDLING_SET_OFF	246
6.9.2.2	PETSC_ERRORHANDLING_SET_ON	246
6.9.2.3	PETSC_FINALIZE	246
6.9.2.4	PETSC_INITIALIZE	247
6.9.2.5	PETSC_ISCOLORINGDESTROY	247
6.9.2.6	PETSC_ISCOLORINGFINALISE	248
6.9.2.7	PETSC_ISCOLORINGINITIALISE	248
6.9.2.8	PETSC_ISDESTROY	248
6.9.2.9	PETSC_ISFINALISE	249
6.9.2.10	PETSC_ISINITIALISE	249
6.9.2.11	PETSC_ISLOCALTOGLOBALMAPPINGAPPLY	249

6.9.2.12 PETSC_ISLOCALTOGLOBALMAPPINGAPPLYIS	250
6.9.2.13 PETSC_ISLOCALTOGLOBALMAPPINGCREATE	250
6.9.2.14 PETSC_ISLOCALTOGLOBALMAPPINGDESTROY	251
6.9.2.15 PETSC_ISLOCALTOGLOBALMAPPINGFINALISE	251
6.9.2.16 PETSC_ISLOCALTOGLOBALMAPPINGINITIALISE	251
6.9.2.17 PETSC_KSPCREATE	252
6.9.2.18 PETSC_KSPDESTROY	252
6.9.2.19 PETSC_KSPFINALISE	252
6.9.2.20 PETSC_KSPGETCONVERGEDREASON	253
6.9.2.21 PETSC_KSPGETITERATIONNUMBER	253
6.9.2.22 PETSC_KSPGETPC	254
6.9.2.23 PETSC_KSPGETRESIDUALNORM	254
6.9.2.24 PETSC_KSPINITIALISE	254
6.9.2.25 PETSC_KSPSETFROMOPTIONS	255
6.9.2.26 PETSC_KSPSETOPERATORS	255
6.9.2.27 PETSC_KSPSETTOLERANCES	256
6.9.2.28 PETSC_KSPSETTYPE	256
6.9.2.29 PETSC_KSPSETUP	257
6.9.2.30 PETSC_KSPSOLVE	257
6.9.2.31 PETSC_LOGPRINTSUMMARY	257
6.9.2.32 PETSC_MATASSEMBLYBEGIN	258
6.9.2.33 PETSC_MATASSEMBLYEND	258
6.9.2.34 PETSC_MATCREATE	258
6.9.2.35 PETSC_MATCREATEMPIAIJ	259
6.9.2.36 PETSC_MATCREATEMPIDENSE	259
6.9.2.37 PETSC_MATCREATESEQAIJ	260
6.9.2.38 PETSC_MATCREATESEQDENSE	260
6.9.2.39 PETSC_MATDESTROY	261
6.9.2.40 PETSC_MATFDCOLORINGCREATE	261
6.9.2.41 PETSC_MATFDCOLORINGDESTROY	262
6.9.2.42 PETSC_MATFDCOLORINGFINALISE	262
6.9.2.43 PETSC_MATFDCOLORINGINITIALISE	262
6.9.2.44 PETSC_MATFDCOLORINGSETFROMOPTIONS	263
6.9.2.45 PETSC_MATFINALISE	263
6.9.2.46 PETSC_MATGETARRAY	264
6.9.2.47 PETSC_MATGETCOLORING	264

6.9.2.48 PETSC_MATGETOWNERSHIPRANGE	264
6.9.2.49 PETSC_MATGETVALUES	265
6.9.2.50 PETSC_MATINITIALISE	265
6.9.2.51 PETSC_MATRESTOREARRAY	266
6.9.2.52 PETSC_MATSETLOCALTOGLOBALMAPPING	266
6.9.2.53 PETSC_MATSETOPTION	266
6.9.2.54 PETSC_MATSETSIZES	267
6.9.2.55 PETSC_MATSETVALUE	267
6.9.2.56 PETSC_MATSETVALUELOCAL	268
6.9.2.57 PETSC_MATSETVALUES	268
6.9.2.58 PETSC_MATSETVALUESLOCAL	269
6.9.2.59 PETSC_MATVIEW	269
6.9.2.60 PETSC_MATZEROENTRIES	269
6.9.2.61 PETSC_PCFINALISE	270
6.9.2.62 PETSC_PCINITIALISE	270
6.9.2.63 PETSC_PCSETTYPE	271
6.9.2.64 PETSC_SNESCREATE	271
6.9.2.65 PETSC_SNESDESTROY	271
6.9.2.66 PETSC_SNESFINALISE	272
6.9.2.67 PETSC_SNESGETCONVERGEDREASON	272
6.9.2.68 PETSC_SNESGETFUNCTIONNORM	273
6.9.2.69 PETSC_SNESGETITERATIONNUMBER	273
6.9.2.70 PETSC_SNESINITIALISE	273
6.9.2.71 PETSC_SNESLINESEARCHSET	274
6.9.2.72 PETSC_SNESLINESEARCHSETPARAMS	274
6.9.2.73 PETSC_SNESSETFROMOPTIONS	275
6.9.2.74 PETSC_SNESSETFUNCTION	275
6.9.2.75 PETSC_SNESSETJACOBIAN_MATFDCOLORING	276
6.9.2.76 PETSC_SNESSETJACOBIAN_SOLVER	276
6.9.2.77 PETSC_SNESSETTOLERANCES	277
6.9.2.78 PETSC_SNESSETTRUSTREGIONTOLERANCE	277
6.9.2.79 PETSC_SNESSETTYPE	278
6.9.2.80 PETSC_SNESSOLVE	278
6.9.2.81 PETSC_VECASSEMBLYBEGIN	278
6.9.2.82 PETSC_VECASSEMBLYEND	279
6.9.2.83 PETSC_VECCREATE	279

6.9.2.84 PETSC_VECCREATEGHOST	280
6.9.2.85 PETSC_VECCREATEGHOSTWITHARRAY	280
6.9.2.86 PETSC_VECCREATEMPI	281
6.9.2.87 PETSC_VECCREATEMPIWITHARRAY	281
6.9.2.88 PETSC_VECCREATESEQ	281
6.9.2.89 PETSC_VECCREATESEQWITHARRAY	282
6.9.2.90 PETSC_VECDESTROY	282
6.9.2.91 PETSC_VECDUPLICATE	283
6.9.2.92 PETSC_VECFINALISE	283
6.9.2.93 PETSC_VECGETARRAY	283
6.9.2.94 PETSC_VECGETARRAYF90	284
6.9.2.95 PETSC_VECGETLOCALSIZE	284
6.9.2.96 PETSC_VECGETOWNERSHIPRANGE	284
6.9.2.97 PETSC_VECGETSIZE	285
6.9.2.98 PETSC_VECGETVALUES	285
6.9.2.99 PETSC_VECHOSTGETLOCALFORM	286
6.9.2.100 PETSC_VECHOSTRESTORELOCALFORM	286
6.9.2.101 PETSC_VECHOSTUPDATEBEGIN	286
6.9.2.102 PETSC_VECHOSTUPDATEEND	287
6.9.2.103 PETSC_VECINITIALISE	287
6.9.2.104 PETSC_VECRESTOREARRAY	287
6.9.2.105 PETSC_VECRESTOREARRAYF90	288
6.9.2.106 PETSC_VECSET	288
6.9.2.107 PETSC_VECSETFROMOPTIONS	289
6.9.2.108 PETSC_VECSETLOCALTOGLOBALMAPPING	289
6.9.2.109 PETSC_VECSETSIZES	289
6.9.2.110 PETSC_VECSETVALUES	290
6.9.2.111 PETSC_VECSETVALUESLOCAL	290
6.9.2.112 PETSC_VECVIEW	291
6.9.3 Variable Documentation	291
6.9.3.1 PETSC_HANDLE_ERROR	291
6.9.3.2 PETSC_SNES_LINESEARCH_CUBIC	291
6.9.3.3 PETSC_SNES_LINESEARCH_NO	291
6.9.3.4 PETSC_SNES_LINESEARCH_NONORMS	291
6.9.3.5 PETSC_SNES_LINESEARCH_QUADRATIC	292
6.10 CMISS_PETSC_TYPES Namespace Reference	293

6.10.1	Detailed Description	293
6.11	COMP_ENVIRONMENT Namespace Reference	294
6.11.1	Detailed Description	295
6.11.2	Function Documentation	295
6.11.2.1	COMPUTATIONAL_ENVIRONMENT_FINALISE	295
6.11.2.2	COMPUTATIONAL_ENVIRONMENT_INITIALISE	295
6.11.2.3	COMPUTATIONAL_NODE_FINALISE	296
6.11.2.4	COMPUTATIONAL_NODE_INITIALISE	296
6.11.2.5	COMPUTATIONAL_NODE_MPI_TYPE_FINALISE	297
6.11.2.6	COMPUTATIONAL_NODE_MPI_TYPE_INITIALISE	297
6.11.2.7	COMPUTATIONAL_NODE_NUMBER_GET	298
6.11.2.8	COMPUTATIONAL_NODES_NUMBER_GET	298
6.11.3	Variable Documentation	299
6.11.3.1	COMPUTATIONAL_ENVIRONMENT	299
6.11.3.2	MPI_COMPUTATIONAL_NODE_TYPE_DATA	299
6.12	COORDINATE_ROUTINES Namespace Reference	300
6.12.1	Detailed Description	302
6.12.2	Function Documentation	302
6.12.2.1	CO	302
6.12.2.2	CO	303
6.12.2.3	COORDINATE_CONVERT_FROM_RC_DP	303
6.12.2.4	COORDINATE_CONVERT_FROM_RC_SP	303
6.12.2.5	COORDINATE_CONVERT_TO_RC_DP	304
6.12.2.6	COORDINATE_CONVERT_TO_RC_SP	304
6.12.2.7	COORDINATE_DELTA_CALCULATE_DP	304
6.12.2.8	COORDINATE_DERIVATIVE_NORM	304
6.12.2.9	COORDINATE_INTERPOLATION_ADJUST	305
6.12.2.10	COORDINATE_INTERPOLATION_PARAMETERS_ADJUST	305
6.12.2.11	COORDINATE_METRICS_CALCULATE	306
6.12.2.12	COORDINATE_SYSTEM_CREATE_FINISH	306
6.12.2.13	COORDINATE_SYSTEM_CREATE_START	306
6.12.2.14	COORDINATE_SYSTEM_DESTROY_NUMBER	306
6.12.2.15	COORDINATE_SYSTEM_DESTROY_PTR	307
6.12.2.16	COORDINATE_SYSTEM_DIMENSION_GET	307
6.12.2.17	COORDINATE_SYSTEM_DIMENSION_SET_NUMBER	307
6.12.2.18	COORDINATE_SYSTEM_DIMENSION_SET_PTR	307

6.12.2.19 COORDINATE_SYSTEM_FOCUS_GET	308
6.12.2.20 COORDINATE_SYSTEM_FOCUS_SET_NUMBER	308
6.12.2.21 COORDINATE_SYSTEM_FOCUS_SET_PTR	308
6.12.2.22 COORDINATE_SYSTEM_NORMAL_CALCULATE	308
6.12.2.23 COORDINATE_SYSTEM_ORIENTATION_SET_NUMBER	309
6.12.2.24 COORDINATE_SYSTEM_ORIENTATION_SET_PTR	309
6.12.2.25 COORDINATE_SYSTEM_ORIGIN_SET_NUMBER	309
6.12.2.26 COORDINATE_SYSTEM_ORIGIN_SET_PTR	309
6.12.2.27 COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_GET	310
6.12.2.28 COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_- SET_NUMBER	310
6.12.2.29 COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_- SET_PTR	310
6.12.2.30 COORDINATE_SYSTEM_TYPE_GET	310
6.12.2.31 COORDINATE_SYSTEM_TYPE_SET_NUMBER	311
6.12.2.32 COORDINATE_SYSTEM_TYPE_SET_PTR	311
6.12.2.33 COORDINATE_SYSTEM_USER_NUMBER_FIND	311
6.12.2.34 COORDINATE_SYSTEMS_FINALISE	312
6.12.2.35 COORDINATE_SYSTEMS_INITIALISE	312
6.12.2.36 D2ZX_DP	312
6.12.2.37 DXZ_DP	312
6.12.2.38 DZX_DP	313
6.12.3 Variable Documentation	313
6.12.3.1 _SYSTEM_TYPE_STRING	313
6.12.3.2 COORDINAT	313
6.12.3.3 COORDINATE_SYSTEMS	313
6.12.3.4 GLOBAL_COORDINATE_SYSTEM	313
6.13 DOMAIN_MAPPINGS Namespace Reference	314
6.13.1 Detailed Description	314
6.13.2 Function Documentation	315
6.13.2.1 DOMAIN_MAPPINGS_ADJACENT_DOMAIN_FINALISE	315
6.13.2.2 DOMAIN_MAPPINGS_ADJACENT_DOMAIN_INITIALISE	315
6.13.2.3 DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE . .	315
6.13.2.4 DOMAIN_MAPPINGS_MAPPING_FINALISE	316
6.13.2.5 DOMAIN_MAPPINGS_MAPPING_GLOBAL_FINALISE	316
6.13.2.6 DOMAIN_MAPPINGS_MAPPING_GLOBAL_INITIALISE	317
6.13.2.7 DOMAIN_MAPPINGS_MAPPING_INITIALISE	317

6.14 ELASTICITY_ROUTINES Namespace Reference	318
6.14.1 Detailed Description	318
6.14.2 Function Documentation	318
6.14.2.1 EL	318
6.14.2.2 ELASTICITY_EQUATIONS_SET_SETUP	319
6.14.2.3 ELASTICITYFINITE_ELEMENT_CALCULATE	319
6.14.2.4 ELASTICITYFINITE_ELEMENT_JACOBIAN_EVALUATE	320
6.14.2.5 ELASTICITYFINITE_ELEMENT_RESIDUAL_EVALUATE	320
6.14.2.6 ELASTICITY_PROBLEM_CLASS_TYPE_SET	321
6.14.2.7 ELASTICITY_PROBLEM_SETUP	321
6.15 ELECTROMECHANICS_ROUTINES Namespace Reference	323
6.15.1 Detailed Description	323
6.16 EQUATIONS_MAPPING_ROUTINES Namespace Reference	324
6.16.1 Detailed Description	326
6.16.2 Function Documentation	326
6.16.2.1 EQUATIONS_MAPPING_CALCULATE	326
6.16.2.2 EQUATIONS_MAPPING_CREATE_FINISH	327
6.16.2.3 EQUATIONS_MAPPING_CREATE_START	327
6.16.2.4 EQUATIONS_MAPPING_CREATE_VALUES_CACHE_FINALISE	328
6.16.2.5 EQUATIONS_MAPPING_CREATE_VALUES_CACHE_INITIALISE	328
6.16.2.6 EQUATIONS_MAPPING_DESTROY	329
6.16.2.7 EQUATIONS_MAPPING_EQUATIONS_JACOBIAN_TO_- VARIABLE_MAP_FINALISE	329
6.16.2.8 EQUATIONS_MAPPING_EQUATIONS_JACOBIAN_TO_- VARIABLE_MAP_INITIALISE	329
6.16.2.9 EQUATIONS_MAPPING_EQUATIONS_MATRIX_TO_- VARIABLE_MAP_FINALISE	330
6.16.2.10 EQUATIONS_MAPPING_EQUATIONS_MATRIX_TO_- VARIABLE_MAP_INITIALISE	330
6.16.2.11 EQUATIONS_MAPPING_FINALISE	331
6.16.2.12 EQUATIONS_MAPPING_INITIALISE	331
6.16.2.13 EQUATIONS_MAPPING_LINEAR_MAPPING_FINALISE	331
6.16.2.14 EQUATIONS_MAPPING_LINEAR_MAPPING_INITIALISE	332
6.16.2.15 EQUATIONS_MAPPING_LINEAR_MATRICES_COEFFICIENTS_- SET	332
6.16.2.16 EQUATIONS_MAPPING_LINEAR_MATRICES_NUMBER_SET	333
6.16.2.17 EQUATIONS_MAPPING_LINEAR_MATRICES_VARIABLE_- TYPES_SET	333

6.16.2.18 EQUATIONS_MAPPING_NONLINEAR_MAPPING_FINALISE	334
6.16.2.19 EQUATIONS_MAPPING_NONLINEAR_MAPPING_INITIALISE	334
6.16.2.20 EQUATIONS_MAPPING_RESIDUAL_COEFFICIENT_SET	334
6.16.2.21 EQUATIONS_MAPPING_RESIDUAL_VARIABLE_TYPE_SET	335
6.16.2.22 EQUATIONS_MAPPING_RHS_COEFFICIENT_SET	335
6.16.2.23 EQUATIONS_MAPPING_RHS_MAPPING_FINALISE	336
6.16.2.24 EQUATIONS_MAPPING_RHS_MAPPING_INITIALISE	336
6.16.2.25 EQUATIONS_MAPPING_RHS_VARIABLE_TYPE_SET	336
6.16.2.26 EQUATIONS_MAPPING_SOURCE_COEFFICIENT_SET	337
6.16.2.27 EQUATIONS_MAPPING_SOURCE_MAPPING_FINALISE	337
6.16.2.28 EQUATIONS_MAPPING_SOURCE_MAPPING_INITIALISE	338
6.16.2.29 EQUATIONS_MAPPING_SOURCE_VARIABLE_TYPE_SET	338
6.16.2.30 EQUATIONS_MAPPING_VARIABLE_TO_EQUATIONS_- COLUMN_MAP_FINALISE	339
6.16.2.31 EQUATIONS_MAPPING_VARIABLE_TO_EQUATIONS_- JACOBIAN_MAP_FINALISE	339
6.16.2.32 EQUATIONS_MAPPING_VARIABLE_TO_EQUATIONS_- JACOBIAN_MAP_INITIALISE	339
6.16.2.33 EQUATIONS_MAPPING_VARIABLE_TO_EQUATIONS_- MATRICES_MAP_FINALISE	340
6.16.2.34 EQUATIONS_MAPPING_VARIABLE_TO_EQUATIONS_- MATRICES_MAP_INITIALISE	340
6.17 EQUATIONS_SET_CONSTANTS Namespace Reference	341
6.17.1 Detailed Description	344
6.17.2 Variable Documentation	344
6.17.2.1 EQUATIONS_SET_ADVECTION_DIFFUSION_EQUATION_TYPE .	344
6.17.2.2 EQUATIONS_SET_BIHAMONIC_EQUATION_TYPE	344
6.17.2.3 EQUATIONS_SET_CLASSICAL_FIELD_CLASS	344
6.17.2.4 EQUATIONS_SET_CONSTANT_SOURCE_POISSON_SUBTYPE .	345
6.17.2.5 EQUATIONS_SET_DIFFUSION_EQUATION_TYPE	345
6.17.2.6 EQUATIONS_SET_ELASTICITY_CLASS	345
6.17.2.7 EQUATIONS_SET_ELECTROMAGNETICS_CLASS	345
6.17.2.8 EQUATIONS_SET_ELECTROSTATIC_TYPE	345
6.17.2.9 EQUATIONS_SETFINITE_ELASTICITY_TYPE	345
6.17.2.10 EQUATIONS_SET_FITTING_CLASS	346
6.17.2.11 EQUATIONS_SET_FLUID_MECHANICS_CLASS	346
6.17.2.12 EQUATIONS_SET_GENERALISED_LAPLACE_SUBTYPE . .	346
6.17.2.13 EQUATIONS_SET_HELMHOLTZ_EQUATION_TYPE	346

6.17.2.14 EQUATIONS_SET_LAPLACE_EQUATION_TYPE	346
6.17.2.15 EQUATIONS_SET_LINEAR_ELASTIC_MODAL_TYPE	346
6.17.2.16 EQUATIONS_SET_LINEAR_ELASTICITY_TYPE	347
6.17.2.17 EQUATIONS_SET_LINEAR_SOURCE_POISSON_SUBTYPE	347
6.17.2.18 EQUATIONS_SET_MAGNETOSTATIC_TYPE	347
6.17.2.19 EQUATIONS_SET_MAXWELLS_EQUATIONS_TYPE	347
6.17.2.20 EQUATIONS_SET_MODAL_CLASS	347
6.17.2.21 EQUATIONS_SET_NAVIER_STOKES_FLUID_TYPE	347
6.17.2.22 EQUATIONS_SET_NO_CLASS	347
6.17.2.23 EQUATIONS_SET_NO_SUBTYPE	347
6.17.2.24 EQUATIONS_SET_NO_TYPE	348
6.17.2.25 EQUATIONS_SET_OPTIMISATION_CLASS	348
6.17.2.26 EQUATIONS_SET_POISSON_EQUATION_TYPE	348
6.17.2.27 EQUATIONS_SET_QUADRATIC_SOURCE_POISSON_SUBTYPE . .	348
6.17.2.28 EQUATIONS_SETREACTION_DIFFUSION_EQUATION_TYPE . .	348
6.17.2.29 EQUATIONS_SET_STANDARD_LAPLACE_SUBTYPE	348
6.17.2.30 EQUATIONS_SET_STOKES_FLUID_TYPE	349
6.17.2.31 EQUATIONS_SET_WAVE_EQUATION_TYPE	349
6.18 F90C Namespace Reference	350
6.18.1 Detailed Description	350
6.18.2 Function Documentation	350
6.18.2.1 C2FSTRING	350
6.18.2.2 CSTRINGLENGTH	351
6.18.2.3 F2CSTRING	351
6.18.2.4 FSTRINGLENGTH	351
6.19 FIELD_IO_ROUTINES Namespace Reference	352
6.19.1 Detailed Description	355
6.19.2 Function Documentation	356
6.19.2.1 FIE	356
6.19.2.2 FIELD_IO_BASIS_LHTP_FAMILY_LABEL	356
6.19.2.3 FIELD_IO_CREATE_FIELDS	356
6.19.2.4 FIELD_IO_DERIVATIVE_INFO	357
6.19.2.5 FIELD_IO_ELEMENTAL_INFO_SET_ATTACH_LOCAL_PROCESS	357
6.19.2.6 FIELD_IO_ELEMENTAL_INFO_SET_FINALIZE	357
6.19.2.7 FIELD_IO_ELEMENTAL_INFO_SET_INITIALISE	358
6.19.2.8 FIELD_IO_ELEMENTAL_INFO_SET_SORT	358

6.19.2.9	FIELD_IO_ELEMENTS_EXPORT	359
6.19.2.10	FIELD_IO_EXPORT_ELEMENTAL_GROUP_HEADER_FORTRAN	359
6.19.2.11	FIELD_IO_EXPORT_ELEMENTS_INTO_	360
6.19.2.12	FIELD_IO_EXPORT_NODAL_GROUP_HEADER_FORTRA	360
6.19.2.13	FIELD_IO_EXPORT_NODES_INTO_LOC	361
6.19.2.14	FIELD_IO_FIELD_INFO	361
6.19.2.15	FIELD_IO_FILEDS_GROUP_INFO_GET	361
6.19.2.16	FIELD_IO_FILEDS_IMPORT	362
6.19.2.17	FIELD_IO_FILL_BASIS_INFO	362
6.19.2.18	FIELD_IO_FORTRAN_FILE_CLOSE	363
6.19.2.19	FIELD_IO_FORTRAN_FILE_OPEN	363
6.19.2.20	FIELD_IO_FORTRAN_FILE_READ_DP	364
6.19.2.21	FIELD_IO_FORTRAN_FILE_READ_INTG	364
6.19.2.22	FIELD_IO_FORTRAN_FILE_READ_STRING	364
6.19.2.23	FIELD_IO_FORTRAN_FILE_REWIND	365
6.19.2.24	FIELD_IO_FORTRAN_FILE_WRITE_DP	365
6.19.2.25	FIELD_IO_FORTRAN_FILE_WRITE_INTG	366
6.19.2.26	FIELD_IO_FORTRAN_FILE_WRITE_STRING	366
6.19.2.27	FIELD_IO_IMPORT_GLOBAL_MESH	367
6.19.2.28	FIELD_IO_LABEL_DERIVATIVE_INFO_GET	367
6.19.2.29	FIELD_IO_LABEL_FIELD_INFO_GET	368
6.19.2.30	FIELD_IO_MULTI_FILES_INFO_GET	368
6.19.2.31	FIELD_IO_NODAL_INFO_SET_ATTACH_LOCAL_PROCESS	368
6.19.2.32	FIELD_IO_NODAL_INFO_SET_FINALIZE	369
6.19.2.33	FIELD_IO_NODAL_INFO_SET_INITIALISE	369
6.19.2.34	FIELD_IO_NODAL_INFO_SET_SORT	370
6.19.2.35	FIELD_IO_NODES_EXPORT	370
6.19.2.36	FIELD_IO_TRANSLATE_LABEL_INTO_INTERPOLATION_TYPE	370
6.19.2.37	STRING_TO_MUTI_INTEGERS_VS	371
6.19.2.38	STRING_TO_MUTI_REALS_VS	371
6.19.3	Variable Documentation	371
6.19.3.1	FIELD_IO_COMPONENT_LABEL	371
6.19.3.2	FIELD_IO_DERIVATIVE_LABEL	371
6.19.3.3	FIELD_IO_FIELD_LABEL	372
6.19.3.4	FIELD_IO_SCALE_FACTORS_NUMBER_TYPE	372
6.19.3.5	FIELD_IO_SCALE_FACTORS_PROPERTY_TYPE	372

6.19.3.6 FIELD_IO_VARIABLE_LABEL	372
6.19.3.7 SHAPE_SIZE	372
6.20 FIELD_ROUTINES Namespace Reference	373
6.20.1 Detailed Description	378
6.20.2 Function Documentation	378
6.20.2.1 FI	378
6.20.2.2 FIELD_COMPONENT_INTER	378
6.20.2.3 FIELD_COMPONENT_INTERPOLATION_GET	379
6.20.2.4 FIELD_COMPONENT_MESH_COM	379
6.20.2.5 FIELD_COMPONENT_MESH_COMPONENT_GET	380
6.20.2.6 FIELD_CREATE_FINISH	380
6.20.2.7 FIELD_CREATE_VALUES_CACHE_FINALISE	381
6.20.2.8 FIELD_CREATE_VALUES_CACHE_INITIALISE	381
6.20.2.9 FIELD_DEPENDENT_TYPE_GET	382
6.20.2.10 FIELD_DEPENDENT_TYPE_SET_NUMBER	382
6.20.2.11 FIELD_DEPENDENT_TYPE_SET_PTR	383
6.20.2.12 FIELD_DESTROY	383
6.20.2.13 FIELD_DIMENSION_GET	383
6.20.2.14 FIELD_DIMENSION_SET_NUMBER	384
6.20.2.15 FIELD_DIMENSION_SET_PTR	384
6.20.2.16 FIELD_INTERPOLATE_GAUSS	385
6.20.2.17 FIELD_INTERPOLATE_XI	386
6.20.2.18 FIELD_INTERPOLATED_POINT_FINALISE	386
6.20.2.19 FIELD_INTERPOLATED_POINT_INITIALISE	387
6.20.2.20 FIELD_INTERPOLATED_POINT_METRICS_CALCULATE	387
6.20.2.21 FIELD_INTERPOLATED_POINT_METRICS_FINALISE	388
6.20.2.22 FIELD_INTERPOLATED_POINT_METRICS_INITIALISE	388
6.20.2.23 FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET	389
6.20.2.24 FIELD_INTERPOLATION_PARAMETERS_FINALISE	389
6.20.2.25 FIELD_INTERPOLATION_PARAMETERS_INITIALISE	390
6.20.2.26 FIELD_INTERPOLATION_PARAMETERS_LINE_GET	390
6.20.2.27 FIELD_VARIABLE_COMPONENT_FINALISE	391
6.20.2.28 FIELD_VARIABLE_COMPONENT_INITIALISE	392
6.20.2.29 FIELD_VARIABLES_FINALISE	392
6.20.2.30 FIELD_VARIABLES_INITIALISE	392
6.20.2.31 FIELDS_FINALISE	393

6.20.2.32 FIELDS_INITIALISE	393
6.21 FINITE_ELASTICITY_ROUTINES Namespace Reference	395
6.21.1 Detailed Description	396
6.21.2 Function Documentation	396
6.21.2.1 FINITE_ELASTICITY_EQUATIONS_SET_SETUP	396
6.21.2.2 FINITE_ELASTICITY_EQUATIONS_SET_SUBTYPE_SET	397
6.21.2.3 FINITE_ELASTICITYFINITE_ELEMENT_JACOBIAN_EVALUATE	397
6.21.2.4 FINITE_ELASTICITYFINITE_ELEMENT_RESIDUAL_EVALUATE	398
6.21.2.5 FINITE_ELASTICITY_GAUSS_CAUCHY_TENSOR	398
6.21.2.6 FINITE_ELASTICITY_GAUSS_DEFORMATION_GRADEINT_TENSOR	399
6.21.2.7 FINITE_ELASTICITY_GAUSS_DFDZ	399
6.21.2.8 FINITE_ELASTICITY_GAUSS_DXDNU	400
6.21.2.9 FINITE_ELASTICITY_PROBLEM_SETUP	400
6.21.2.10 FINITE_ELASTICITY_PROBLEM_SUBTYPE_SET	401
6.22 FINITE_ELEMENT_ROUTINES Namespace Reference	402
6.22.1 Detailed Description	402
6.22.2 Function Documentation	402
6.22.2.1 FEM_ELEMENT_MATRICES_FINALISE	402
6.22.2.2 FEM_ELEMENT_MATRICES_INITIALISE	402
6.23 FLUID_MECHANICS_ROUTINES Namespace Reference	403
6.23.1 Detailed Description	403
6.24 GENERATED_MESH_ROUTINES Namespace Reference	404
6.24.1 Detailed Description	406
6.24.2 Function Documentation	406
6.24.2.1 GENERATED_MESH_BASIS_GET	406
6.24.2.2 GENERATED_MESH_BASIS_SET_NUMBER	407
6.24.2.3 GENERATED_MESH_BASIS_SET_PTR	407
6.24.2.4 GENERATED_MESH_CREATE_FINISH	408
6.24.2.5 GENERATED_MESH_CREATE_START	408
6.24.2.6 GENERATED_MESH_DESTROY_NUMBER	408
6.24.2.7 GENERATED_MESH_DESTROY_PTR	409
6.24.2.8 GENERATED_MESH_EXTENT_GET	409
6.24.2.9 GENERATED_MESH_EXTENT_SET_NUMBER	410
6.24.2.10 GENERATED_MESH_EXTENT_SET_PTR	410
6.24.2.11 GENERATED_MESH_FINALISE	410
6.24.2.12 GENERATED_MESH_INITIALISE	411

6.24.2.13	GENERATED_MESH_NUMBER_OF_ELEMENTS_GET	411
6.24.2.14	GENERATED_MESH_NUMBER_OF_ELEMENTS_SET_NUMBER . .	411
6.24.2.15	GENERATED_MESH_NUMBER_OF_ELEMENTS_SET_PTR	412
6.24.2.16	GENERATED_MESH_ORIGIN_GET	412
6.24.2.17	GENERATED_MESH_ORIGIN_SET_NUMBER	412
6.24.2.18	GENERATED_MESH_ORIGIN_SET_PTR	413
6.24.2.19	GENERATED_MESH_REGULAR_CREATE_FINISH	413
6.24.2.20	GENERATED_MESH_REGULAR_FINALISE	414
6.24.2.21	GENERATED_MESH_REGULAR_INITIALISE	414
6.24.2.22	GENERATED_MESH_TYPE_GET	415
6.24.2.23	GENERATED_MESH_TYPE_SET_NUMBER	415
6.24.2.24	GENERATED_MESH_TYPE_SET_PTR	415
6.24.2.25	GENERATED_MESH_USER_NUMBER_FIND	416
6.24.2.26	GENERATED_MESHES_FINALISE	416
6.24.2.27	GENERATED_MESHES_INITIALISE	417
6.24.3	Variable Documentation	417
6.24.3.1	GENERATED_MESHES	417
6.25	INPUT_OUTPUT Namespace Reference	418
6.25.1	Detailed Description	431
6.25.2	Function Documentation	432
6.25.2.1	WR	432
6.25.2.2	WR	432
6.25.2.3	WR	433
6.25.2.4	WR	434
6.25.2.5	WR	435
6.25.2.6	WR	435
6.25.2.7	WR	436
6.25.2.8	WR	437
6.25.2.9	WR	437
6.25.2.10	WR	438
6.25.2.11	WR	438
6.25.2.12	WR	439
6.25.2.13	WR	440
6.25.2.14	WR	440
6.25.2.15	WR	441
6.25.2.16	WR	441

6.25.2.17 WR	442
6.25.2.18 WR	443
6.25.2.19 WR	443
6.25.2.20 WR	444
6.25.2.21 WR	444
6.25.2.22 WR	445
6.25.2.23 WR	446
6.25.2.24 WR	446
6.25.2.25 WR	447
6.25.2.26 WR	447
6.25.2.27 WR	448
6.25.2.28 WR	449
6.25.2.29 WR	449
6.25.2.30 WR	450
6.25.2.31 WR	450
6.25.2.32 WR	451
6.25.2.33 WR	452
6.25.2.34 WR	452
6.25.2.35 WRITE_STRING_C	453
6.25.2.36 WRITE_STRING_FMT	453
6.25.2.37 WRITE_STRING_FMT	454
6.25.2.38 WRITE_STRING_FMT	455
6.25.2.39 WRITE_STRING_FMT	455
6.25.2.40 WRITE_STRING_FMT	456
6.25.2.41 WRITE_STRING_FMT	457
6.25.2.42 WRITE_STRING_FMT	457
6.25.2.43 WRITE_STRING_FMT_VALUE_C	458
6.25.2.44 WRITE_STRING_FMT_VALUE_DP	458
6.25.2.45 WRITE_STRING_FMT_VALUE_INTG	459
6.25.2.46 WRITE_STRING_FMT_VALUE_L	459
6.25.2.47 WRITE_STRING_FMT_VALUE_LINTG	460
6.25.2.48 WRITE_STRING_FMT_VALUE_SP	460
6.25.2.49 WRITE_STRING_FMT_VALUE_VS	461
6.25.2.50 WRITE_STRING_IDX	462
6.25.2.51 WRITE_STRING_IDX	462
6.25.2.52 WRITE_STRING_IDX	463

6.25.2.53 WRITE_STRING_IDX	464
6.25.2.54 WRITE_STRING_IDX	465
6.25.2.55 WRITE_STRING_MATRIX_DP	466
6.25.2.56 WRITE_STRING_MATRIX_INTG	467
6.25.2.57 WRITE_STRING_MATRIX_L	468
6.25.2.58 WRITE_STRING_MATRIX_LINTG	469
6.25.2.59 WRITE_STRING_MATRIX_SP	470
6.25.2.60 WRITE_STRING_TWO_VALUE_C_C	472
6.25.2.61 WRITE_STRING_TWO_VALUE_C_DP	472
6.25.2.62 WRITE_STRING_TWO_VALUE_C_INTG	473
6.25.2.63 WRITE_STRING_TWO_VALUE_C_L	473
6.25.2.64 WRITE_STRING_TWO_VALUE_C_SP	474
6.25.2.65 WRITE_STRING_TWO_VALUE_C_VS	475
6.25.2.66 WRITE_STRING_TWO_VALUE_DP_C	475
6.25.2.67 WRITE_STRING_TWO_VALUE_DP_DP	476
6.25.2.68 WRITE_STRING_TWO_VALUE_DP_INTG	476
6.25.2.69 WRITE_STRING_TWO_VALUE_DP_L	477
6.25.2.70 WRITE_STRING_TWO_VALUE_DP_SP	477
6.25.2.71 WRITE_STRING_TWO_VALUE_DP_VS	478
6.25.2.72 WRITE_STRING_TWO_VALUE_INTG_C	479
6.25.2.73 WRITE_STRING_TWO_VALUE_INTG_DP	479
6.25.2.74 WRITE_STRING_TWO_VALUE_INTG_INTG	480
6.25.2.75 WRITE_STRING_TWO_VALUE_INTG_L	480
6.25.2.76 WRITE_STRING_TWO_VALUE_INTG_SP	481
6.25.2.77 WRITE_STRING_TWO_VALUE_INTG_VS	482
6.25.2.78 WRITE_STRING_TWO_VALUE_L_C	482
6.25.2.79 WRITE_STRING_TWO_VALUE_L_DP	483
6.25.2.80 WRITE_STRING_TWO_VALUE_L_INTG	483
6.25.2.81 WRITE_STRING_TWO_VALUE_L_L	484
6.25.2.82 WRITE_STRING_TWO_VALUE_L_SP	484
6.25.2.83 WRITE_STRING_TWO_VALUE_L_VS	485
6.25.2.84 WRITE_STRING_TWO_VALUE_SP_C	486
6.25.2.85 WRITE_STRING_TWO_VALUE_SP_DP	486
6.25.2.86 WRITE_STRING_TWO_VALUE_SP_INTG	487
6.25.2.87 WRITE_STRING_TWO_VALUE_SP_L	487
6.25.2.88 WRITE_STRING_TWO_VALUE_SP_SP	488

6.25.2.89 WRITE_STRING_TWO_VALUE_SP_VS	489
6.25.2.90 WRITE_STRING_TWO_VALUE_VS_C	489
6.25.2.91 WRITE_STRING_TWO_VALUE_VS_DP	490
6.25.2.92 WRITE_STRING_TWO_VALUE_VS_INTG	490
6.25.2.93 WRITE_STRING_TWO_VALUE_VS_L	491
6.25.2.94 WRITE_STRING_TWO_VALUE_VS_SP	492
6.25.2.95 WRITE_STRING_TWO_VALUE_VS_VS	492
6.25.2.96 WRITE_STRING_VALUE_C	493
6.25.2.97 WRITE_STRING_VALUE_DP	493
6.25.2.98 WRITE_STRING_VALUE_INTG	494
6.25.2.99 WRITE_STRING_VALUE_L	494
6.25.2.100 WRITE_STRING_VALUE_LINTG	495
6.25.2.101 WRITE_STRING_VALUE_SP	495
6.25.2.102 WRITE_STRING_VALUE_VS	496
6.25.2.103 WRITE_STRING_VS	496
6.26 ISO_VARYING_STRING Namespace Reference	497
6.26.1 Detailed Description	499
6.26.2 Function Documentation	499
6.26.2.1 adjustl_	499
6.26.2.2 adjustr_	500
6.26.2.3 char_auto	500
6.26.2.4 char_fixed	500
6.26.2.5 extract_CH	500
6.26.2.6 extract_VS	500
6.26.2.7 get_	500
6.26.2.8 get_set_CH	500
6.26.2.9 get_set_VS	500
6.26.2.10 get_unit	501
6.26.2.11 get_unit_set_CH	501
6.26.2.12 get_unit_set_VS	501
6.26.2.13 iachar_	501
6.26.2.14 ichar_	501
6.26.2.15 index_CH_VS	501
6.26.2.16 index_VS_CH	501
6.26.2.17 index_VS_VS	501
6.26.2.18 insert_CH_CH	502

6.26.2.19 insert_CH_VS	502
6.26.2.20 insert_VS_CH	502
6.26.2.21 insert_VS_VS	502
6.26.2.22 len_	502
6.26.2.23 len_trim_	502
6.26.2.24 lge_CH_VS	502
6.26.2.25 lge_VS_CH	502
6.26.2.26 lge_VS_VS	502
6.26.2.27 lgt_CH_VS	503
6.26.2.28 lgt_VS_CH	503
6.26.2.29 lgt_VS_VS	503
6.26.2.30 lle_CH_VS	503
6.26.2.31 lle_VS_CH	503
6.26.2.32 lle_VS_VS	503
6.26.2.33 llt_CH_VS	503
6.26.2.34 llt_VS_CH	503
6.26.2.35 llt_VS_VS	503
6.26.2.36 op_assign_CH_VS	504
6.26.2.37 op_assign_VS_CH	504
6.26.2.38 op_concat_CH_VS	504
6.26.2.39 op_concat_VS_CH	504
6.26.2.40 op_concat_VS_VS	504
6.26.2.41 op_eq_CH_VS	504
6.26.2.42 op_eq_VS_CH	504
6.26.2.43 op_eq_VS_VS	505
6.26.2.44 op_ge_CH_VS	505
6.26.2.45 op_ge_VS_CH	505
6.26.2.46 op_ge_VS_VS	505
6.26.2.47 op_gt_CH_VS	505
6.26.2.48 op_gt_VS_CH	505
6.26.2.49 op_gt_VS_VS	505
6.26.2.50 op_le_CH_VS	505
6.26.2.51 op_le_VS_CH	505
6.26.2.52 op_le_VS_VS	506
6.26.2.53 op_lt_CH_VS	506
6.26.2.54 op_lt_VS_CH	506

6.26.2.55 op_lt_VS_VS	506
6.26.2.56 op_ne_CH_VS	506
6.26.2.57 op_ne_VS_CH	506
6.26.2.58 op_ne_VS_VS	506
6.26.2.59 put_CH	506
6.26.2.60 put_line_CH	506
6.26.2.61 put_line_unit_CH	507
6.26.2.62 put_line_unit_VS	507
6.26.2.63 put_line_VS	507
6.26.2.64 put_unit_CH	507
6.26.2.65 put_unit_VS	507
6.26.2.66 put_VS	507
6.26.2.67 remove_CH	507
6.26.2.68 remove_VS	507
6.26.2.69 repeat_	507
6.26.2.70 replace_CH_CH_auto	508
6.26.2.71 replace_CH_CH_CH_target	508
6.26.2.72 replace_CH_CH_fixed	508
6.26.2.73 replace_CH_CH_VS_target	508
6.26.2.74 replace_CH_VS_auto	508
6.26.2.75 replace_CH_VS_CH_target	508
6.26.2.76 replace_CH_VS_fixed	508
6.26.2.77 replace_CH_VS_VS_target	509
6.26.2.78 replace_VS_CH_auto	509
6.26.2.79 replace_VS_CH_CH_target	509
6.26.2.80 replace_VS_CH_fixed	509
6.26.2.81 replace_VS_CH_VS_target	509
6.26.2.82 replace_VS_VS_auto	509
6.26.2.83 replace_VS_VS_CH_target	509
6.26.2.84 replace_VS_VS_fixed	510
6.26.2.85 replace_VS_VS_VS_target	510
6.26.2.86 scan_CH_VS	510
6.26.2.87 scan_VS_CH	510
6.26.2.88 scan_VS_VS	510
6.26.2.89 split_CH	510
6.26.2.90 split_VS	510

6.26.2.91 trim_	511
6.26.2.92 var_str_	511
6.26.2.93 verify_CH_VS	511
6.26.2.94 verify_VS_CH	511
6.26.2.95 verify_VS_VS	511
6.26.3 Variable Documentation	511
6.26.3.1 GET_BUFFER_LEN	511
6.27 KINDS Namespace Reference	512
6.27.1 Detailed Description	512
6.28 LAPACK Namespace Reference	513
6.28.1 Detailed Description	513
6.29 LAPLACE_EQUATIONS_ROUTINES Namespace Reference	514
6.29.1 Detailed Description	515
6.30 LINEAR_ELASTICITY_ROUTINES Namespace Reference	516
6.30.1 Detailed Description	516
6.30.2 Function Documentation	516
6.30.2.1 LINEAR_ELASTICITY_EQUATIONS_SET_SETUP	516
6.30.2.2 LINEAR_ELASTICITY_EQUATIONS_SET_SUBTYPE_SET	517
6.30.2.3 LINEAR_ELASTICITYFINITE_ELEMENT_CALCULATE	518
6.30.2.4 LINEAR_ELASTICITY_PROBLEM_SETUP	518
6.30.2.5 LINEAR_ELASTICITY_PROBLEM_SUBTYPE_SET	519
6.31 LISTS Namespace Reference	520
6.31.1 Detailed Description	524
6.31.2 Function Documentation	524
6.31.2.1 LIST_CREATE_FINISH	524
6.31.2.2 LIST_CREATE_START	524
6.31.2.3 LIST_DATA_TYPE_SET	525
6.31.2.4 LIST_DESTROY	525
6.31.2.5 LIST_DETACH_AND_DESTROY_DP	526
6.31.2.6 LIST_DETACH_AND_DESTROY_INTG	526
6.31.2.7 LIST_DETACH_AND_DESTROY_SP	527
6.31.2.8 LIST_FINALISE	527
6.31.2.9 LIST_INITIAL_SIZE_SET	528
6.31.2.10 LIST_INITIALISE	528
6.31.2.11 LIST_ITEM_ADD_DP1	529
6.31.2.12 LIST_ITEM_ADD_INTG1	529

6.31.2.13 LIST_ITEM_ADD_SP1	529
6.31.2.14 LIST_ITEM_DELETE	530
6.31.2.15 LIST_ITEM_IN_LIST_DP1	530
6.31.2.16 LIST_ITEM_IN_LIST_INTG1	531
6.31.2.17 LIST_ITEM_IN_LIST_SP1	531
6.31.2.18 LIST_NUMBER_OF_ITEMS_GET	531
6.31.2.19 LIST_REMOVE_DUPLICATES	532
6.31.2.20 LIST_SEARCH_DP_ARRAY	532
6.31.2.21 LIST_SEARCH_INTG_ARRAY	533
6.31.2.22 LIST_SEARCH_LINEAR_DP_ARRAY	533
6.31.2.23 LIST_SEARCH_LINEAR_INTG_ARRAY	534
6.31.2.24 LIST_SEARCH_LINEAR_SP_ARRAY	534
6.31.2.25 LIST_SEARCH_SP_ARRAY	535
6.31.2.26 LIST_SORT_BUBBLE_DP_ARRAY	535
6.31.2.27 LIST_SORT_BUBBLE_INTG_ARRAY	535
6.31.2.28 LIST_SORT_BUBBLE_SP_ARRAY	536
6.31.2.29 LIST_SORT_DP_ARRAY	536
6.31.2.30 LIST_SORT_HEAP_DP_ARRAY	537
6.31.2.31 LIST_SORT_HEAP_INTG_ARRAY	537
6.31.2.32 LIST_SORT_HEAP_SP_ARRAY	537
6.31.2.33 LIST_SORT_INTG_ARRAY	538
6.31.2.34 LIST_SORT_SHELL_DP_ARRAY	538
6.31.2.35 LIST_SORT_SHELL_INTG_ARRAY	538
6.31.2.36 LIST_SORT_SHELL_SP_ARRAY	539
6.31.2.37 LIST_SORT_SP_ARRAY	539
6.32 MACHINE_CONSTANTS Namespace Reference	540
6.32.1 Detailed Description	540
6.32.2 Variable Documentation	540
6.32.2.1 CHARACTER_SIZE	540
6.32.2.2 DOUBLE_COMPLEX_SIZE	540
6.32.2.3 DOUBLE_REAL_SIZE	541
6.32.2.4 ERROR_SEPARATOR_CONSTANT	541
6.32.2.5 INTEGER_SIZE	541
6.32.2.6 LOGICAL_SIZE	541
6.32.2.7 LONG_INTEGER_SIZE	541
6.32.2.8 MACHINE_CHAR_FORMAT	541

6.32.2.9 MACHINE_DP_FORMAT	541
6.32.2.10 MACHINE_ENDIAN	541
6.32.2.11 MACHINE_INT_FORMAT	542
6.32.2.12 MACHINE_OS	542
6.32.2.13 MACHINE_SP_FORMAT	542
6.32.2.14 MACHINE_TYPE	542
6.32.2.15 SHORT_INTEGER_SIZE	542
6.32.2.16 SINGLE_COMPLEX_SIZE	542
6.32.2.17 SINGLE_REAL_SIZE	543
6.33 MATHS Namespace Reference	544
6.33.1 Detailed Description	545
6.33.2 Function Documentation	545
6.33.2.1 CROSS_PRODUCT_DP	545
6.33.2.2 CROSS_PRODUCT_INTG	545
6.33.2.3 CROSS_PRODUCT_SP	545
6.33.2.4 D_CROSS_PRODUCT_DP	546
6.33.2.5 D_CROSS_PRODUCT_INTG	546
6.33.2.6 D_CROSS_PRODUCT_SP	546
6.33.2.7 DETERMINANT_FULL_DP	546
6.33.2.8 DETERMINANT_FULL_INTG	547
6.33.2.9 DETERMINANT_FULL_SP	547
6.33.2.10 EDP_DP	547
6.33.2.11 EDP_SP	547
6.33.2.12 EIGENVALUE_FULL_DP	547
6.33.2.13 EIGENVALUE_FULL_SP	547
6.33.2.14 EIGENVECTOR_FULL_DP	548
6.33.2.15 EIGENVECTOR_FULL_SP	548
6.33.2.16 IO_DP	548
6.33.2.17 IO_SP	548
6.33.2.18 II_DP	548
6.33.2.19 II_SP	548
6.33.2.20 INVERT_FULL_DP	549
6.33.2.21 INVERT_FULL_SP	549
6.33.2.22 K0_DP	549
6.33.2.23 K0_SP	549
6.33.2.24 K1_DP	549

6.33.2.25 K1_SP	549
6.33.2.26 KDP_DP	549
6.33.2.27 KDP_SP	550
6.33.2.28 L2NORM_DP	550
6.33.2.29 L2NORM_SP	550
6.33.2.30 MATRIX_PRODUCT_DP	550
6.33.2.31 MATRIX_PRODUCT_SP	550
6.33.2.32 MATRIX_TRANSPOSE_DP	550
6.33.2.33 MATRIX_TRANSPOSE_SP	551
6.33.2.34 NORMALISE_DP	551
6.33.2.35 NORMALISE_SP	551
6.33.2.36 SOLVE_SMALL_LINEAR_SYSTEM_DP	551
6.33.2.37 SOLVE_SMALL_LINEAR_SYSTEM_SP	551
6.34 MATRIX_VECTOR Namespace Reference	552
6.34.1 Detailed Description	560
6.34.2 Function Documentation	560
6.34.2.1 MATRIX_ALL_VALUES_SET_DP	560
6.34.2.2 MATRIX_ALL_VALUES_SET_INTG	560
6.34.2.3 MATRIX_ALL_VALUES_SET_L	561
6.34.2.4 MATRIX_ALL_VALUES_SET_SP	561
6.34.2.5 MATRIX_CREATE_FINISH	562
6.34.2.6 MATRIX_CREATE_START	562
6.34.2.7 MATRIX_DATA_GET_DP	562
6.34.2.8 MATRIX_DATA_GET_INTG	563
6.34.2.9 MATRIX_DATA_GET_L	563
6.34.2.10 MATRIX_DATA_GET_SP	564
6.34.2.11 MATRIX_DATA_TYPE_SET	564
6.34.2.12 MATRIX_DESTROY	565
6.34.2.13 MATRIX_DUPLICATE	565
6.34.2.14 MATRIX_FINALISE	565
6.34.2.15 MATRIX_INITIALISE	566
6.34.2.16 MATRIX_MAX_COLUMNS_PER_ROW_GET	566
6.34.2.17 MATRIX_MAX_SIZE_SET	567
6.34.2.18 MATRIX_NUMBER_NON_ZEROS_SET	567
6.34.2.19 MATRIX_OUTPUT	568
6.34.2.20 MATRIX_SIZE_SET	568

6.34.2.21 MATRIX_STORAGE_LOCATION_FIND	569
6.34.2.22 MATRIX_STORAGE_LOCATIONS_GET	569
6.34.2.23 MATRIX_STORAGE_LOCATIONS_SET	570
6.34.2.24 MATRIX_STORAGE_TYPE_GET	570
6.34.2.25 MATRIX_STORAGE_TYPE_SET	571
6.34.2.26 MATRIX_VALUES_ADD_DP	571
6.34.2.27 MATRIX_VALUES_ADD_DP1	572
6.34.2.28 MATRIX_VALUES_ADD_DP2	572
6.34.2.29 MATRIX_VALUES_ADD_INTG	573
6.34.2.30 MATRIX_VALUES_ADD_INTG1	573
6.34.2.31 MATRIX_VALUES_ADD_INTG2	574
6.34.2.32 MATRIX_VALUES_ADD_L	574
6.34.2.33 MATRIX_VALUES_ADD_L1	575
6.34.2.34 MATRIX_VALUES_ADD_L2	575
6.34.2.35 MATRIX_VALUES_ADD_SP	576
6.34.2.36 MATRIX_VALUES_ADD_SP1	576
6.34.2.37 MATRIX_VALUES_ADD_SP2	577
6.34.2.38 MATRIX_VALUES_GET_DP	577
6.34.2.39 MATRIX_VALUES_GET_DP1	578
6.34.2.40 MATRIX_VALUES_GET_DP2	578
6.34.2.41 MATRIX_VALUES_GET_INTG	579
6.34.2.42 MATRIX_VALUES_GET_INTG1	579
6.34.2.43 MATRIX_VALUES_GET_INTG2	580
6.34.2.44 MATRIX_VALUES_GET_L	580
6.34.2.45 MATRIX_VALUES_GET_L1	581
6.34.2.46 MATRIX_VALUES_GET_L2	581
6.34.2.47 MATRIX_VALUES_GET_SP	582
6.34.2.48 MATRIX_VALUES_GET_SP1	582
6.34.2.49 MATRIX_VALUES_GET_SP2	583
6.34.2.50 MATRIX_VALUES_SET_DP	583
6.34.2.51 MATRIX_VALUES_SET_DP1	584
6.34.2.52 MATRIX_VALUES_SET_DP2	584
6.34.2.53 MATRIX_VALUES_SET_INTG	585
6.34.2.54 MATRIX_VALUES_SET_INTG1	585
6.34.2.55 MATRIX_VALUES_SET_INTG2	586
6.34.2.56 MATRIX_VALUES_SET_L	586

6.34.2.57 MATRIX_VALUES_SET_L1	587
6.34.2.58 MATRIX_VALUES_SET_L2	587
6.34.2.59 MATRIX_VALUES_SET_SP	588
6.34.2.60 MATRIX_VALUES_SET_SP1	588
6.34.2.61 MATRIX_VALUES_SET_SP2	589
6.34.2.62 VECTOR_ALL_VALUES_SET_DP	589
6.34.2.63 VECTOR_ALL_VALUES_SET_INTG	590
6.34.2.64 VECTOR_ALL_VALUES_SET_L	590
6.34.2.65 VECTOR_ALL_VALUES_SET_SP	590
6.34.2.66 VECTOR_CREATE_FINISH	591
6.34.2.67 VECTOR_CREATE_START	591
6.34.2.68 VECTOR_DATA_GET_DP	592
6.34.2.69 VECTOR_DATA_GET_INTG	592
6.34.2.70 VECTOR_DATA_GET_L	592
6.34.2.71 VECTOR_DATA_GET_SP	593
6.34.2.72 VECTOR_DATA_TYPE_SET	593
6.34.2.73 VECTOR_DESTROY	594
6.34.2.74 VECTOR_DUPLICATE	594
6.34.2.75 VECTOR_FINALISE	595
6.34.2.76 VECTOR_INITIALISE	595
6.34.2.77 VECTOR_SIZE_SET	595
6.34.2.78 VECTOR_VALUES_GET_DP	596
6.34.2.79 VECTOR_VALUES_GET_DP1	596
6.34.2.80 VECTOR_VALUES_GET_INTG	597
6.34.2.81 VECTOR_VALUES_GET_INTG1	597
6.34.2.82 VECTOR_VALUES_GET_L	597
6.34.2.83 VECTOR_VALUES_GET_L1	598
6.34.2.84 VECTOR_VALUES_GET_SP	598
6.34.2.85 VECTOR_VALUES_GET_SP1	599
6.34.2.86 VECTOR_VALUES_SET_DP	599
6.34.2.87 VECTOR_VALUES_SET_DP1	599
6.34.2.88 VECTOR_VALUES_SET_INTG	600
6.34.2.89 VECTOR_VALUES_SET_INTG1	600
6.34.2.90 VECTOR_VALUES_SET_L	601
6.34.2.91 VECTOR_VALUES_SET_L1	601
6.34.2.92 VECTOR_VALUES_SET_SP	601

6.34.2.93 VECTOR_VALUES_SET_SP1	602
6.34.3 Variable Documentation	602
6.34.3.1 MATRIX_VECTOR_ID	602
6.35 MESH_ROUTINES Namespace Reference	603
6.35.1 Detailed Description	608
6.35.2 Function Documentation	608
6.35.2.1 DECOMPOSITION_CREATE_FINISH	608
6.35.2.2 DECOMPOSITION_CREATE_START	609
6.35.2.3 DECOMPOSITION_DESTROY	609
6.35.2.4 DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE	610
6.35.2.5 DECOMPOSITION_ELEMENT_DOMAIN_GET	610
6.35.2.6 DECOMPOSITION_ELEMENT_DOMAIN_SET	611
6.35.2.7 DECOMPOSITION_MESH_COMPONENT_NUMBER_GET	611
6.35.2.8 DECOMPOSITION_MESH_COMPONENT_NUMBER_SET	612
6.35.2.9 DECOMPOSITION_NUMBER_OF_DOMAINS_GET	612
6.35.2.10 DECOMPOSITION_NUMBER_OF_DOMAINS_SET	612
6.35.2.11 DOMAIN_MAPPINGS_NODES_FINALISE	613
6.35.2.12 DOMAIN_MAPPINGS_NODES_INITIALISE	614
6.35.2.13 DOMAIN_TOPOLOGY_CALCULATE	614
6.35.2.14 DOMAIN_TOPOLOGY_DOFS_FINALISE	614
6.35.2.15 DOMAIN_TOPOLOGY_DOFS_INITIALISE	615
6.35.2.16 DOMAIN_TOPOLOGY_ELEMENT_FINALISE	615
6.35.2.17 DOMAIN_TOPOLOGY_ELEMENT_INITIALISE	616
6.35.2.18 DOMAIN_TOPOLOGY_ELEMENTS_FINALISE	616
6.35.2.19 DOMAIN_TOPOLOGY_ELEMENTS_INITIALISE	617
6.35.2.20 DOMAIN_TOPOLOGY_FINALISE	617
6.35.2.21 DOMAIN_TOPOLOGY_INITIALISE	618
6.35.2.22 DOMAIN_TOPOLOGY_INITIALISE_FROM_MESH	618
6.35.2.23 DOMAIN_TOPOLOGY_LINE_FINALISE	619
6.35.2.24 DOMAIN_TOPOLOGY_LINE_INITIALISE	619
6.35.2.25 DOMAIN_TOPOLOGY_LINES_FINALISE	619
6.35.2.26 DOMAIN_TOPOLOGY_LINES_INITIALISE	620
6.35.2.27 DOMAIN_TOPOLOGY_NODE_FINALISE	620
6.35.2.28 DOMAIN_TOPOLOGY_NODE_INITIALISE	621
6.35.2.29 DOMAIN_TOPOLOGY_NODES_FINALISE	621
6.35.2.30 DOMAIN_TOPOLOGY_NODES_INITIALISE	622

6.35.2.31 DOMAIN_TOPOLOGY_NODES_SURROUNDING_ELEMENTS_- CALCULATE	622
6.35.2.32 MESH_CREATE_FINISH	623
6.35.2.33 MESH_CREATE_START	623
6.35.2.34 MESH_DESTROY	624
6.35.2.35 MESH_FINALISE	624
6.35.2.36 MESH_INITIALISE	625
6.35.2.37 MESH_NUMBER_OF_COMPONENTS_GET	625
6.35.2.38 MESH_NUMBER_OF_COMPONENTS_SET_NUMBER	625
6.35.2.39 MESH_NUMBER_OF_COMPONENTS_SET_PTR	626
6.35.2.40 MESH_NUMBER_OF_ELEMENTS_GET	626
6.35.2.41 MESH_NUMBER_OF_ELEMENTS_SET_NUMBER	627
6.35.2.42 MESH_NUMBER_OF_ELEMENTS_SET_PTR	627
6.35.2.43 MESH_TOPOLOGY_CALCULATE	627
6.35.2.44 MESH_TOPOLOGY_DOFS_CALCULATE	628
6.35.2.45 MESH_TOPOLOGY_DOFS_FINALISE	628
6.35.2.46 MESH_TOPOLOGY_DOFS_INITIALISE	629
6.35.2.47 MESH_TOPOLOGY_ELEMENT_FINALISE	629
6.35.2.48 MESH_TOPOLOGY_ELEMENT_INITIALISE	630
6.35.2.49 MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_- CALCULATE	630
6.35.2.50 MESH_TOPOLOGY_ELEMENTS_BASIS_SET	631
6.35.2.51 MESH_TOPOLOGY_ELEMENTS_CREATE_FINISH	631
6.35.2.52 MESH_TOPOLOGY_ELEMENTS_CREATE_START	632
6.35.2.53 MESH_TOPOLOGY_ELEMENTS_DESTROY	632
6.35.2.54 MESH_TOPOLOGY_ELEMENTS_ELEMENT_BASIS_GET	633
6.35.2.55 MESH_TOPOLOGY_ELEMENTS_ELEMENT_BASIS_SET	633
6.35.2.56 MESH_TOPOLOGY_ELEMENTS_ELEMENT_NODES_GET	634
6.35.2.57 MESH_TOPOLOGY_ELEMENTS_ELEMENT_NODES_SET	634
6.35.2.58 MESH_TOPOLOGY_ELEMENTS_FINALISE	635
6.35.2.59 MESH_TOPOLOGY_ELEMENTS_INITIALISE	635
6.35.2.60 MESH_TOPOLOGY_ELEMENTS_NUMBER_GET	636
6.35.2.61 MESH_TOPOLOGY_ELEMENTS_NUMBER_SET	636
6.35.2.62 MESH_TOPOLOGY_FINALISE	637
6.35.2.63 MESH_TOPOLOGY_INITIALISE	637
6.35.2.64 MESH_TOPOLOGY_NODE_FINALISE	638
6.35.2.65 MESH_TOPOLOGY_NODE_INITIALISE	638

6.35.2.66 MESH_TOPOLOGY_NODES_CALCULATE	639
6.35.2.67 MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE	639
6.35.2.68 MESH_TOPOLOGY_NODES_FINALISE	640
6.35.2.69 MESH_TOPOLOGY_NODES_INITIALISE	640
6.35.2.70 MESH_TOPOLOGY_NODES_SURROUNDING_ELEMENTS_- CALCULATE	641
6.35.2.71 MESH_USER_NUMBER_FIND	641
6.35.2.72 MESHES_FINALISE	642
6.35.2.73 MESHES_INITIALISE	642
6.36 NODE_ROUTINES Namespace Reference	643
6.36.1 Detailed Description	643
6.36.2 Function Documentation	643
6.36.2.1 NODE_CHECK_EXISTS	643
6.36.2.2 NODE_DESTROY	643
6.36.2.3 NODE_INITIAL_POSITION_SET	644
6.36.2.4 NODE_NUMBER_SET	644
6.36.2.5 NODES_CREATE_FINISH	644
6.36.2.6 NODES_CREATE_START	645
6.36.2.7 NODES_FINALISE	645
6.36.2.8 NODES_INITIALISE	645
6.37 PROBLEM_CONSTANTS Namespace Reference	646
6.37.1 Detailed Description	647
6.37.2 Variable Documentation	647
6.37.2.1 PROBLEM_ADVECTION_DIFFUSION_EQUATION_TYPE	647
6.37.2.2 PROBLEM_BIHAMONIC_EQUATION_TYPE	648
6.37.2.3 PROBLEM_CLASSICAL_FIELD_CLASS	648
6.37.2.4 PROBLEM_DIFFUSION_EQUATION_TYPE	648
6.37.2.5 PROBLEM_ELASTICITY_CLASS	648
6.37.2.6 PROBLEM_ELECTROMAGNETICS_CLASS	648
6.37.2.7 PROBLEM_ELECTROSTATIC_TYPE	648
6.37.2.8 PROBLEMFINITE_ELASTICITY_TYPE	649
6.37.2.9 PROBLEM_FITTING_CLASS	649
6.37.2.10 PROBLEM_FLUID_MECHANICS_CLASS	649
6.37.2.11 PROBLEM_GENERALISED_LAPLACE_SUBTYPE	649
6.37.2.12 PROBLEM_HELMHOLTZ_EQUATION_TYPE	649
6.37.2.13 PROBLEM_LAPLACE_EQUATION_TYPE	649
6.37.2.14 PROBLEM_LINEAR_ELASTIC_MODAL_TYPE	650

6.37.2.15 PROBLEM_LINEAR_ELASTICITY_TYPE	650
6.37.2.16 PROBLEM_MAGNETOSTATIC_TYPE	650
6.37.2.17 PROBLEM_MAXWELLS_EQUATIONS_TYPE	650
6.37.2.18 PROBLEM_MODAL_CLASS	650
6.37.2.19 PROBLEM_NAVIER_STOKES_FLUID_TYPE	650
6.37.2.20 PROBLEM_NO_CLASS	650
6.37.2.21 PROBLEM_NO_SUBTYPE	650
6.37.2.22 PROBLEM_NO_TYPE	651
6.37.2.23 PROBLEM_OPTIMISATION_CLASS	651
6.37.2.24 PROBLEM_POISSON_EQUATION_TYPE	651
6.37.2.25 PROBLEMREACTION_DIFFUSION_EQUATION_TYPE	651
6.37.2.26 PROBLEM_STANDARD_LAPLACE_SUBTYPE	651
6.37.2.27 PROBLEM_STOKES_FLUID_TYPE	651
6.37.2.28 PROBLEM_WAVE_EQUATION_TYPE	651
6.38 PROBLEM_ROUTINES Namespace Reference	652
6.38.1 Detailed Description	655
6.38.2 Function Documentation	655
6.38.2.1 PROBLEM_CONTROL_CREATE_FINISH	655
6.38.2.2 PROBLEM_CONTROL_CREATE_START	655
6.38.2.3 PROBLEM_CONTROL_DESTROY	656
6.38.2.4 PROBLEM_CONTROL_FINALISE	656
6.38.2.5 PROBLEM_CONTROL_INITIALISE	656
6.38.2.6 PROBLEM_CREATE_FINISH	657
6.38.2.7 PROBLEM_CREATE_START	657
6.38.2.8 PROBLEM_DESTROY_NUMBER	658
6.38.2.9 PROBLEM_DESTROY_PTR	658
6.38.2.10 PROBLEM_FINALISE	658
6.38.2.11 PROBLEM_INITIALISE	659
6.38.2.12 PROBLEM_SETUP	659
6.38.2.13 PROBLEM_SOLUTION_EQUATIONS_SET_ADD	660
6.38.2.14 PROBLEM_SOLUTION_FINALISE	661
6.38.2.15 PROBLEM_SOLUTION_INITIALISE	661
6.38.2.16 PROBLEM_SOLUTION_JACOBIAN_EVALUATE	661
6.38.2.17 PROBLEM_SOLUTION_RESIDUAL_EVALUATE	662
6.38.2.18 PROBLEM_SOLUTION_SOLVE	662
6.38.2.19 PROBLEM_SOLUTIONS_CREATE_FINISH	663

6.38.2.20 PROBLEM_SOLUTIONS_CREATE_START	663
6.38.2.21 PROBLEM_SOLUTIONS_FINALISE	664
6.38.2.22 PROBLEM_SOLUTIONS_INITIALISE	664
6.38.2.23 PROBLEM_SOLVE	664
6.38.2.24 PROBLEM_SOLVER_CREATE_FINISH	665
6.38.2.25 PROBLEM_SOLVER_CREATE_START	665
6.38.2.26 PROBLEM_SOLVER_DESTROY	666
6.38.2.27 PROBLEM_SOLVER_GET	666
6.38.2.28 PROBLEM_SPECIFICATION_GET_NUMBER	666
6.38.2.29 PROBLEM_SPECIFICATION_GET_PTR	667
6.38.2.30 PROBLEM_SPECIFICATION_SET_NUMBER	667
6.38.2.31 PROBLEM_SPECIFICATION_SET_PTR	668
6.38.2.32 PROBLEM_USER_NUMBER_FIND	668
6.38.2.33 PROBLEMS_FINALISE	669
6.38.2.34 PROBLEMS_INITIALISE	669
6.38.3 Variable Documentation	670
6.38.3.1 PROBLEMS	670
6.39 REGION_ROUTINES Namespace Reference	671
6.39.1 Detailed Description	671
6.39.2 Function Documentation	671
6.39.2.1 REGION_COORDINATE_SYSTEM_GET	671
6.39.2.2 REGION_COORDINATE_SYSTEM_SET_NUMBER	672
6.39.2.3 REGION_COORDINATE_SYSTEM_SET_PTR	672
6.39.2.4 REGION_CREATE_FINISH	672
6.39.2.5 REGION_CREATE_START	672
6.39.2.6 REGION_DESTROY	673
6.39.2.7 REGION_LABEL_GET	673
6.39.2.8 REGION_LABEL_SET_NUMBER	673
6.39.2.9 REGION_LABEL_SET_PTR	673
6.39.2.10 REGION_SUB_REGION_CREATE_FINISH	674
6.39.2.11 REGION_SUB_REGION_CREATE_START	674
6.39.2.12 REGION_USER_NUMBER_FIND	674
6.39.2.13 REGION_USER_NUMBER_FIND_PTR	675
6.39.2.14 REGIONS_FINALISE	675
6.39.2.15 REGIONS_INITIALISE	675
6.39.3 Variable Documentation	675

6.39.3.1	GLOBAL_REGION	675
6.40	SOLVER_MATRICES_ROUTINES Namespace Reference	676
6.40.1	Detailed Description	677
6.40.2	Function Documentation	677
6.40.2.1	SOLVER_MATRICES_CREATE_FINISH	677
6.40.2.2	SOLVER_MATRICES_CREATE_START	677
6.40.2.3	SOLVER_MATRICES_DESTROY	678
6.40.2.4	SOLVER_MATRICES_FINALISE	678
6.40.2.5	SOLVER_MATRICES_INITIALISE	679
6.40.2.6	SOLVER_MATRICES_LIBRARY_TYPE_GET	679
6.40.2.7	SOLVER_MATRICES_LIBRARY_TYPE_SET	679
6.40.2.8	SOLVER_MATRICES_OUTPUT	680
6.40.2.9	SOLVER_MATRICES_STORAGE_TYPE_GET	680
6.40.2.10	SOLVER_MATRICES_STORAGE_TYPE_SET	681
6.40.2.11	SOLVER_MATRIX_FINALISE	681
6.40.2.12	SOLVER_MATRIX_FORM	682
6.40.2.13	SOLVER_MATRIX_INITIALISE	682
6.40.2.14	SOLVER_MATRIX_STRUCTURE_CALCULATE	683
6.41	SOLVER_ROUTINES Namespace Reference	684
6.41.1	Detailed Description	691
6.41.2	Function Documentation	691
6.41.2.1	SOLVER_CREATE_FINISH	691
6.41.2.2	SOLVER_CREATE_START	692
6.41.2.3	SOLVER_DESTROY	692
6.41.2.4	SOLVER_EIGENPROBLEM_CREATE_FINISH	693
6.41.2.5	SOLVER_EIGENPROBLEM_FINALISE	693
6.41.2.6	SOLVER_EIGENPROBLEM_INITIALISE	694
6.41.2.7	SOLVER_EIGENPROBLEM_SOLVE	694
6.41.2.8	SOLVER_FINALISE	694
6.41.2.9	SOLVER_INITIALISE	695
6.41.2.10	SOLVER_LIBRARY_SET	695
6.41.2.11	SOLVER_LINEAR_CREATE_FINISH	696
6.41.2.12	SOLVER_LINEAR_DIRECT_CREATE_FINISH	697
6.41.2.13	SOLVER_LINEAR_DIRECT_FINALISE	697
6.41.2.14	SOLVER_LINEAR_DIRECT_INITIALISE	698
6.41.2.15	SOLVER_LINEAR_DIRECT_SOLVE	698

6.41.2.16 SOLVER_LINEAR_DIRECT_TYPE_SET	698
6.41.2.17 SOLVER_LINEAR_FINALISE	699
6.41.2.18 SOLVER_LINEAR_INITIALISE	699
6.41.2.19 SOLVER_LINEAR_ITERATIVE_ABSOLUTE_TOLERANCE_SET	700
6.41.2.20 SOLVER_LINEAR_ITERATIVE_CREATE_FINISH	700
6.41.2.21 SOLVER_LINEAR_ITERATIVE_DIVERGENCE_TOLERANCE_SET	701
6.41.2.22 SOLVER_LINEAR_ITERATIVE_FINALISE	701
6.41.2.23 SOLVER_LINEAR_ITERATIVE_INITIALISE	702
6.41.2.24 SOLVER_LINEAR_ITERATIVE_MAXIMUM_ITERATIONS_SET	702
6.41.2.25 SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET	703
6.41.2.26 SOLVER_LINEAR_ITERATIVE_RELATIVE_TOLERANCE_SET	703
6.41.2.27 SOLVER_LINEAR_ITERATIVE_SOLVE	704
6.41.2.28 SOLVER_LINEAR_ITERATIVE_TYPE_SET	704
6.41.2.29 SOLVER_LINEAR_SOLVE	705
6.41.2.30 SOLVER_LINEAR_TYPE_SET	705
6.41.2.31 SOLVER_MATRICES_ASSEMBLE	705
6.41.2.32 SOLVER_NONLINEAR_ABSOLUTE_TOLERANCE_SET	706
6.41.2.33 SOLVER_NONLINEAR_CREATE_FINISH	706
6.41.2.34 SOLVER_NONLINEAR_FINALISE	707
6.41.2.35 SOLVER_NONLINEAR_INITIALISE	707
6.41.2.36 SOLVER_NONLINEAR_JACOBIAN_EVALUATE	708
6.41.2.37 SOLVER_NONLINEAR_JACOBIAN_EVALUATE_PETSC	708
6.41.2.38 SOLVER_NONLINEAR_LINESearch_ALPHA_SET	709
6.41.2.39 SOLVER_NONLINEAR_LINESearch_CREATE_FINISH	709
6.41.2.40 SOLVER_NONLINEAR_LINESearch_FINALISE	710
6.41.2.41 SOLVER_NONLINEAR_LINESearch_INITIALISE	710
6.41.2.42 SOLVER_NONLINEAR_LINESearch_MAXSTEP_SET	711
6.41.2.43 SOLVER_NONLINEAR_LINESearch_SOLVE	711
6.41.2.44 SOLVER_NONLINEAR_LINESearch_STEPTOL_SET	712
6.41.2.45 SOLVER_NONLINEAR_LINESearch_TYPE_SET	712
6.41.2.46 SOLVER_NONLINEAR_MAXIMUM_FUNCTION_- EVALUATIONS_SET	712
6.41.2.47 SOLVER_NONLINEAR_MAXIMUM_ITERATIONS_SET	713
6.41.2.48 SOLVER_NONLINEAR_RELATIVE_TOLERANCE_SET	713
6.41.2.49 SOLVER_NONLINEAR_RESIDUAL_EVALUATE	714
6.41.2.50 SOLVER_NONLINEAR_RESIDUAL_EVALUATE_PETSC	714
6.41.2.51 SOLVER_NONLINEAR SOLUTION_TOLERANCE_SET	714

6.41.2.52 SOLVER_NONLINEAR_SOLVE	715
6.41.2.53 SOLVER_NONLINEAR_TRUSTREGION_CREATE_FINISH	715
6.41.2.54 SOLVER_NONLINEAR_TRUSTREGION_DELTA0_SET	716
6.41.2.55 SOLVER_NONLINEAR_TRUSTREGION_FINALISE	716
6.41.2.56 SOLVER_NONLINEAR_TRUSTREGION_INITIALISE	717
6.41.2.57 SOLVER_NONLINEAR_TRUSTREGION_SOLVE	717
6.41.2.58 SOLVER_NONLINEAR_TRUSTREGION_TOLERANCE_SET	717
6.41.2.59 SOLVER_NONLINEAR_TYPE_SET	718
6.41.2.60 SOLVER_OUTPUT_TYPE_SET	718
6.41.2.61 SOLVER_SOLVE	719
6.41.2.62 SOLVER_SPARSITY_TYPE_SET	719
6.41.2.63 SOLVER_TIME_INTEGRATION_CREATE_FINISH	720
6.41.2.64 SOLVER_TIME_INTEGRATION_FINALISE	720
6.41.2.65 SOLVER_TIME_INTEGRATION_INITIALISE	721
6.41.2.66 SOLVER_TIME_INTEGRATION_SOLVE	721
6.41.2.67 SOLVER_VARIABLES_UPDATE	721
6.41.3 Variable Documentation	722
6.41.3.1 SOLVER_4TH_RUNGE_KUTTA_INTEGRATOR	722
6.41.3.2 SOLVER_ADAMS_MOULTON_INTEGRATOR	722
6.41.3.3 SOLVER_EULER_INTEGRATOR	722
6.41.3.4 SOLVER_IMPROVED_EULER_INTEGRATOR	722
6.41.3.5 SOLVER_LSODA_INTEGRATOR	722
6.42 SORTING Namespace Reference	723
6.42.1 Detailed Description	723
6.42.2 Function Documentation	723
6.42.2.1 BUBBLE_SORT_DP	723
6.42.2.2 BUBBLE_SORT_INTG	723
6.42.2.3 BUBBLE_SORT_SP	724
6.42.2.4 HEAP_SORT_DP	724
6.42.2.5 HEAP_SORT_INTG	724
6.42.2.6 HEAP_SORT_SP	724
6.42.2.7 SHELL_SORT_DP	724
6.42.2.8 SHELL_SORT_INTG	725
6.42.2.9 SHELL_SORT_SP	725
6.43 STRINGS Namespace Reference	726
6.43.1 Detailed Description	730

6.43.2 Function Documentation	730
6.43.2.1 CHARACTER_TO_LOWERCase_C	730
6.43.2.2 CHARACTER_TO_LOWERCase_VS	730
6.43.2.3 CHARACTER_TO_UPPERCase_C	731
6.43.2.4 CHARACTER_TO_UPPERCase_VS	731
6.43.2.5 IS_ABBREVIATION_C_C	731
6.43.2.6 IS_ABBREVIATION_C_VS	731
6.43.2.7 IS_ABBREVIATION_VS_C	732
6.43.2.8 IS_ABBREVIATION_VS_VS	732
6.43.2.9 IS_DIGIT	732
6.43.2.10 IS_LETTER	732
6.43.2.11 IS_LOWERCase	733
6.43.2.12 IS_UPPERCase	733
6.43.2.13 IS_WHITESPACE	733
6.43.2.14 LIST_TO_CHARACTER_C	733
6.43.2.15 LIST_TO_CHARACTER_DP	734
6.43.2.16 LIST_TO_CHARACTER_INTG	734
6.43.2.17 LIST_TO_CHARACTER_L	735
6.43.2.18 LIST_TO_CHARACTER_LINTG	735
6.43.2.19 LIST_TO_CHARACTER_SP	736
6.43.2.20 LOGICAL_TO_CHARACTER	736
6.43.2.21 LOGICAL_TO_VSTRING	737
6.43.2.22 NUMBER_TO_CHARACTER_DP	737
6.43.2.23 NUMBER_TO_CHARACTER_INTG	738
6.43.2.24 NUMBER_TO_CHARACTER_LINTG	738
6.43.2.25 NUMBER_TO_CHARACTER_SP	739
6.43.2.26 NUMBER_TO_VSTRING_DP	739
6.43.2.27 NUMBER_TO_VSTRING_INTG	740
6.43.2.28 NUMBER_TO_VSTRING_LINTG	740
6.43.2.29 NUMBER_TO_VSTRING_SP	740
6.43.2.30 STRING_TO_DOUBLE_C	741
6.43.2.31 STRING_TO_DOUBLE_VS	741
6.43.2.32 STRING_TO_INTEGER_C	742
6.43.2.33 STRING_TO_INTEGER_VS	742
6.43.2.34 STRING_TO_LOGICAL_C	742
6.43.2.35 STRING_TO_LOGICAL_VS	743

6.43.2.36 STRING_TO_LONG_INTEGER_C	743
6.43.2.37 STRING_TO_LONG_INTEGER_VS	743
6.43.2.38 STRING_TO_SINGLE_C	744
6.43.2.39 STRING_TO_SINGLE_VS	744
6.43.2.40 VSTRING_TO_LOWERCASE_C	744
6.43.2.41 VSTRING_TO_LOWERCASE_VS	745
6.43.2.42 VSTRING_TO_UPPERCASE_C	745
6.43.2.43 VSTRING_TO_UPPERCASE_VS	745
6.44 TIMER Namespace Reference	746
6.44.1 Detailed Description	746
6.44.2 Function Documentation	746
6.44.2.1 CPU_TIMER	746
6.44.3 Variable Documentation	746
6.44.3.1 SYSTEM_CPU	746
6.44.3.2 TOTAL_CPU	746
6.44.3.3 USER_CPU	747
6.45 TREES Namespace Reference	748
6.45.1 Detailed Description	749
6.45.2 Function Documentation	750
6.45.2.1 TREE_CREATE_FINISH	750
6.45.2.2 TREE_CREATE_START	750
6.45.2.3 TREE_DESTROY	751
6.45.2.4 TREE_DETACH_AND_DESTROY	751
6.45.2.5 TREE_DETACH_IN_ORDER	751
6.45.2.6 TREE_FINALISE	752
6.45.2.7 TREE_INITIALISE	752
6.45.2.8 TREE_INSERT_TYPE_SET	753
6.45.2.9 TREE_ITEM_DELETE	753
6.45.2.10 TREE_ITEM_INSERT	754
6.45.2.11 TREE_NODE_FINALISE	754
6.45.2.12 TREE_NODE_INITIALISE	755
6.45.2.13 TREE_NODE_KEY_GET	755
6.45.2.14 TREE_NODE_VALUE_GET	756
6.45.2.15 TREE_NODE_VALUE_SET	756
6.45.2.16 TREE_OUTPUT	756
6.45.2.17 TREE_OUTPUT_IN_ORDER	757

6.45.2.18 TREE_PREDECESSOR	757
6.45.2.19 TREE_SEARCH	758
6.45.2.20 TREE_SUCCESSOR	758
6.46 TYPES Namespace Reference	760
6.46.1 Detailed Description	768
7 Class Documentation	769
7.1 BASE_ROUTINES::FLAG_ERROR Interface Reference	769
7.1.1 Detailed Description	769
7.1.2 Member Function Documentation	769
7.1.2.1 FLAG_ERROR_C	769
7.1.2.2 FLAG_ERROR_VS	770
7.2 BASE_ROUTINES::FLAG_WARNING Interface Reference	771
7.2.1 Detailed Description	771
7.2.2 Member Function Documentation	771
7.2.2.1 FLAG_WARNING_C	771
7.2.2.2 FLAG_WARNING_VS	771
7.3 BASE_ROUTINES::interface Interface Reference	772
7.3.1 Detailed Description	772
7.3.2 Member Function Documentation	772
7.3.2.1 CPUTIMER	772
7.4 BASE_ROUTINES::ROUTINE_LIST_ITEM_TYPE Struct Reference	773
7.4.1 Detailed Description	773
7.4.2 Member Data Documentation	773
7.4.2.1 NAME	773
7.4.2.2 NEXT_ROUTINE	774
7.4.2.3 NUMBER_OF_INVOCATIONS	774
7.4.2.4 TOTAL_EXCLUSIVE_CPU_TIME	774
7.4.2.5 TOTAL_EXCLUSIVE_SYSTEM_TIME	774
7.4.2.6 TOTAL_INCLUSIVE_CPU_TIME	774
7.4.2.7 TOTAL_INCLUSIVE_SYSTEM_TIME	774
7.5 BASE_ROUTINES::ROUTINE_LIST_TYPE Struct Reference	775
7.5.1 Detailed Description	775
7.5.2 Member Data Documentation	775
7.5.2.1 HEAD	775
7.6 BASE_ROUTINES::ROUTINE_STACK_ITEM_TYPE Struct Reference	776
7.6.1 Detailed Description	776

7.6.2	Member Data Documentation	776
7.6.2.1	DIAGNOSTICS	776
7.6.2.2	EXCLUSIVE_CPU_TIME	777
7.6.2.3	EXCLUSIVE_SYSTEM_TIME	777
7.6.2.4	INCLUSIVE_CPU_TIME	777
7.6.2.5	INCLUSIVE_SYSTEM_TIME	777
7.6.2.6	NAME	777
7.6.2.7	PREVIOUS_ROUTINE	777
7.6.2.8	ROUTINE_LIST_ITEM	777
7.6.2.9	TIMING	777
7.7	BASE_ROUTINES::ROUTINE_STACK_TYPE Struct Reference	779
7.7.1	Detailed Description	779
7.7.2	Member Data Documentation	779
7.7.2.1	STACK_POINTER	779
7.8	BINARY_FILE::BINARY_FILE_INFO_TYPE Struct Reference	780
7.8.1	Detailed Description	780
7.8.2	Member Data Documentation	780
7.8.2.1	ACCESS_TYPE	780
7.8.2.2	BINARY_FILE_REVISION	780
7.8.2.3	CHAR_FORMAT	780
7.8.2.4	CHARACTER_SIZE	780
7.8.2.5	DP_FORMAT	781
7.8.2.6	DP_REAL_SIZE	781
7.8.2.7	DPC_REAL_SIZE	781
7.8.2.8	ENDIAN_TYPE	781
7.8.2.9	FILE_NAME	781
7.8.2.10	FILE_NUMBER	781
7.8.2.11	INT_FORMAT	781
7.8.2.12	INTEGER_SIZE	781
7.8.2.13	LINTEGER_SIZE	781
7.8.2.14	LOGICAL_SIZE	781
7.8.2.15	MACHINE_TYPE	781
7.8.2.16	OS_TYPE	782
7.8.2.17	SINTEGER_SIZE	782
7.8.2.18	SP_FORMAT	782
7.8.2.19	SP_REAL_SIZE	782

7.8.2.20 SPC_REAL_SIZE	782
7.9 BINARY_FILE::BINARY_FILE_TYPE Struct Reference	783
7.9.1 Detailed Description	783
7.9.2 Member Data Documentation	783
7.9.2.1 FILE_INFORMATION	783
7.10 BINARY_FILE::BINARY_TAG_TYPE Struct Reference	784
7.10.1 Detailed Description	784
7.10.2 Member Data Documentation	784
7.10.2.1 HEADER	784
7.10.2.2 INDEX	784
7.10.2.3 NUM_BYTES	784
7.10.2.4 NUM_HEADER_BYTES	784
7.10.2.5 NUM_SUBTAGS	784
7.11 BINARY_FILE::interface Interface Reference	785
7.11.1 Detailed Description	785
7.11.2 Member Function Documentation	785
7.11.2.1 BINARYCLOSEFILE	785
7.11.2.2 BINARYOPENFILE	785
7.11.2.3 BINARYSETFILE	785
7.11.2.4 BINARYSKIPFILE	785
7.11.2.5 ISBINARYFILEOPEN	786
7.11.2.6 ISENDBINARYFILE	786
7.12 BINARY_FILE::READ_BINARY_FILE Interface Reference	787
7.12.1 Detailed Description	787
7.12.2 Member Function Documentation	787
7.12.2.1 READ_BINARY_FILE_CHARACTER	787
7.12.2.2 READ_BINARY_FILE_DP	787
7.12.2.3 READ_BINARY_FILE_DP1	787
7.12.2.4 READ_BINARY_FILE_DPC	788
7.12.2.5 READ_BINARY_FILE_DPC1	788
7.12.2.6 READ_BINARY_FILE_INTG	788
7.12.2.7 READ_BINARY_FILE_INTG1	788
7.12.2.8 READ_BINARY_FILE_LINTG	788
7.12.2.9 READ_BINARY_FILE_LINTG1	788
7.12.2.10 READ_BINARY_FILE_LOGICAL	788
7.12.2.11 READ_BINARY_FILE_LOGICAL1	789

7.12.2.12 READ_BINARY_FILE_SINTG	789
7.12.2.13 READ_BINARY_FILE_SINTG1	789
7.12.2.14 READ_BINARY_FILE_SP	789
7.12.2.15 READ_BINARY_FILE_SP1	789
7.12.2.16 READ_BINARY_FILE_SPC	789
7.12.2.17 READ_BINARY_FILE_SPC1	790
7.13 BINARYFILE::WRITE_BINARYFILE Interface Reference	791
7.13.1 Detailed Description	791
7.13.2 Member Function Documentation	791
7.13.2.1 WRITE_BINARYFILE_CHARACTER	791
7.13.2.2 WRITE_BINARYFILE_DP	791
7.13.2.3 WRITE_BINARYFILE_DP1	791
7.13.2.4 WRITE_BINARYFILE_DPC	792
7.13.2.5 WRITE_BINARYFILE_DPC1	792
7.13.2.6 WRITE_BINARYFILE_INTG	792
7.13.2.7 WRITE_BINARYFILE_INTG1	792
7.13.2.8 WRITE_BINARYFILE_LINTG	792
7.13.2.9 WRITE_BINARYFILE_LINTG1	792
7.13.2.10 WRITE_BINARYFILE_LOGICAL	792
7.13.2.11 WRITE_BINARYFILE_LOGICAL1	793
7.13.2.12 WRITE_BINARYFILE_SINTG	793
7.13.2.13 WRITE_BINARYFILE_SINTG1	793
7.13.2.14 WRITE_BINARYFILE_SP	793
7.13.2.15 WRITE_BINARYFILE_SP1	793
7.13.2.16 WRITE_BINARYFILE_SPC	793
7.13.2.17 WRITE_BINARYFILE_SPC1	794
7.14 BLAS::interface Interface Reference	795
7.14.1 Detailed Description	795
7.14.2 Member Function Documentation	795
7.14.2.1 DASUM	795
7.14.2.2 DAXPY	795
7.14.2.3 DCOPY	795
7.14.2.4 DDOT	796
7.14.2.5 DNRM2	796
7.14.2.6 DROT	796
7.14.2.7 DROTG	796

7.14.2.8	DSCAL	796
7.14.2.9	DTRSV	796
7.14.2.10	SASUM	796
7.14.2.11	SAXPY	796
7.14.2.12	SCOPY	797
7.14.2.13	SDOT	797
7.14.2.14	SNRM2	797
7.14.2.15	SROT	797
7.14.2.16	SROTG	797
7.14.2.17	SSCAL	797
7.14.2.18	STRSV	797
7.15	CMISS_PARMETIS::interface Interface Reference	798
7.15.1	Detailed Description	798
7.15.2	Member Function Documentation	798
7.15.2.1	ParMETIS_V3_PartK	798
7.15.2.2	ParMETIS_V3_PartMeshKway	798
7.16	CMISS_PETSC::interface Interface Reference	799
7.16.1	Detailed Description	801
7.16.2	Member Function Documentation	801
7.16.2.1	ISColoringDestroy	801
7.16.2.2	ISDestroy	801
7.16.2.3	ISLocalToGlobalMappingApply	801
7.16.2.4	ISLocalToGlobalMappingApplyIS	801
7.16.2.5	ISLocalToGlobalMappingCreate	801
7.16.2.6	ISLocalToGlobalMappingDestroy	801
7.16.2.7	KSPCreate	801
7.16.2.8	KSPDestroy	801
7.16.2.9	KSPGetConvergedReason	801
7.16.2.10	KSPGetIterationNumber	802
7.16.2.11	KSPGetPC	802
7.16.2.12	KSPGetResidualNorm	802
7.16.2.13	KSPSetFromOptions	802
7.16.2.14	KSPSetOperators	802
7.16.2.15	KSPSetTolerances	802
7.16.2.16	KSPSetType	802
7.16.2.17	KSPSetUp	802

7.16.2.18 KSPSolve	802
7.16.2.19 MatAssemblyBegin	802
7.16.2.20 MatAssemblyEnd	803
7.16.2.21 MatCreate	803
7.16.2.22 MatCreateMPIAIJ	803
7.16.2.23 MatCreateMPIDense	803
7.16.2.24 MatCreateSeqAIJ	803
7.16.2.25 MatCreateSeqDense	803
7.16.2.26 MatDestroy	803
7.16.2.27 MatFDColoringCreate	803
7.16.2.28 MatFDColoringDestroy	803
7.16.2.29 MatFDColoringSetFromOptions	803
7.16.2.30 MatGetArray	804
7.16.2.31 MatGetArrayF90	804
7.16.2.32 MatGetColoring	804
7.16.2.33 MatGetOwnershipRange	804
7.16.2.34 MatGetValues	804
7.16.2.35 MatRestoreArray	804
7.16.2.36 MatRestoreArrayF90	804
7.16.2.37 MatSetLocalToGlobalMapping	804
7.16.2.38 MatSetOption	804
7.16.2.39 MatSetSizes	804
7.16.2.40 MatSetValue	805
7.16.2.41 MatSetValueLocal	805
7.16.2.42 MatSetValues	805
7.16.2.43 MatSetValuesLocal	805
7.16.2.44 MatView	805
7.16.2.45 MatZeroEntries	805
7.16.2.46 PCSetType	805
7.16.2.47 PetscFinalize	805
7.16.2.48 PetscInitialize	805
7.16.2.49 PetscLogPrintSummary	805
7.16.2.50 SNESCreate	806
7.16.2.51 SNESDestroy	806
7.16.2.52 SNESGetConvergedReason	806
7.16.2.53 SNESGetFunctionNorm	806

7.16.2.54 SNESGetIterationNumber	806
7.16.2.55 SNESLineSearchSet	806
7.16.2.56 SNESLineSearchSetParams	806
7.16.2.57 SNESSetFromOptions	806
7.16.2.58 SNESSetFunction	806
7.16.2.59 SNESSetTolerances	806
7.16.2.60 SNESSetTrustRegionTolerance	807
7.16.2.61 SNESSetType	807
7.16.2.62 SNESSolve	807
7.16.2.63 VecAssemblyBegin	807
7.16.2.64 VecAssemblyEnd	807
7.16.2.65 VecCreate	807
7.16.2.66 VecCreateGhost	807
7.16.2.67 VecCreateGhostWithArray	807
7.16.2.68 VecCreateMPI	807
7.16.2.69 VecCreateMPIWithArray	807
7.16.2.70 VecCreateSeq	808
7.16.2.71 VecCreateSeqWithArray	808
7.16.2.72 VecDestroy	808
7.16.2.73 VecDuplicate	808
7.16.2.74 VecGetArray	808
7.16.2.75 VecGetArrayF90	808
7.16.2.76 VecGetLocalSize	808
7.16.2.77 VecGetOwnershipRange	808
7.16.2.78 VecGetSize	808
7.16.2.79 VecGetValues	808
7.16.2.80 VecGhostGetLocalForm	809
7.16.2.81 VecGhostRestoreLocalForm	809
7.16.2.82 VecGhostUpdateBegin	809
7.16.2.83 VecGhostUpdateEnd	809
7.16.2.84 VecRestoreArray	809
7.16.2.85 VecRestoreArrayF90	809
7.16.2.86 VecSet	809
7.16.2.87 VecSetFromOptions	809
7.16.2.88 VecSetLocalToGlobalMapping	809
7.16.2.89 VecSetSizes	809

7.16.2.90	VecSetValues	810
7.16.2.91	VecSetValuesLocal	810
7.16.2.92	VecView	810
7.17	CMISS_PETSC::PETSC_SNESSETJACOBIAN Interface Reference	811
7.17.1	Detailed Description	811
7.17.2	Member Function Documentation	811
7.17.2.1	PETSC_SNESSETJACOBIAN_MATFDCOLORING	811
7.17.2.2	PETSC_SNESSETJACOBIAN_SOLVER	811
7.18	CMISS_PETSC_TYPES::PETSC_IS_TYPE Struct Reference	812
7.18.1	Detailed Description	812
7.19	CMISS_PETSC_TYPES::PETSC_ISCOLORING_TYPE Struct Reference	813
7.19.1	Detailed Description	813
7.20	CMISS_PETSC_TYPES::PETSC_ISLOCALTOGLOBALMAPPING_TYPE Struct Reference	814
7.20.1	Detailed Description	814
7.21	CMISS_PETSC_TYPES::PETSC_KSP_TYPE Struct Reference	815
7.21.1	Detailed Description	815
7.22	CMISS_PETSC_TYPES::PETSC_MAT_TYPE Struct Reference	816
7.22.1	Detailed Description	816
7.23	CMISS_PETSC_TYPES::PETSC_MATFDCOLORING_TYPE Struct Reference	817
7.23.1	Detailed Description	817
7.24	CMISS_PETSC_TYPES::PETSC_PC_TYPE Struct Reference	818
7.24.1	Detailed Description	818
7.25	CMISS_PETSC_TYPES::PETSC_SNES_TYPE Struct Reference	819
7.25.1	Detailed Description	819
7.26	CMISS_PETSC_TYPES::PETSC_VEC_TYPE Struct Reference	820
7.26.1	Detailed Description	820
7.27	COMP_ENVIRONMENT::CACHE_TYPE Struct Reference	821
7.27.1	Detailed Description	821
7.27.2	Member Data Documentation	821
7.27.2.1	NUMBER_LEVELS	821
7.27.2.2	SIZE	821
7.28	COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_TYPE Struct Reference	822
7.28.1	Detailed Description	822
7.28.2	Member Data Documentation	822
7.28.2.1	COMPUTATIONAL_NODES	822

7.28.2.2	MPI_COMM	822
7.28.2.3	MY_COMPUTATIONAL_NODE_NUMBER	822
7.28.2.4	NUMBER_COMPUTATIONAL_NODES	823
7.29	COMP_ENVIRONMENT::COMPUTATIONAL_NODE_TYPE Struct Reference	824
7.29.1	Detailed Description	824
7.29.2	Member Data Documentation	824
7.29.2.1	NODE_NAME	824
7.29.2.2	NODE_NAME_LENGTH	824
7.29.2.3	NUMBER_PROCESSORS	824
7.29.2.4	RANK	825
7.30	COMP_ENVIRONMENT::MPI_COMPUTATIONAL_NODE_TYPE Struct Reference	826
7.30.1	Detailed Description	826
7.30.2	Member Data Documentation	826
7.30.2.1	BLOCK_LENGTHS	826
7.30.2.2	DISPLACEMENTS	826
7.30.2.3	MPI_TYPE	826
7.30.2.4	NUM_BLOCKS	827
7.30.2.5	TYPES	827
7.31	COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC Interface Reference	828
7.31.1	Detailed Description	828
7.31.2	Member Function Documentation	828
7.31.2.1	COORDINATE_CONVERT_FROM_RC_DP	828
7.31.2.2	COORDINATE_CONVERT_FROM_RC_SP	828
7.32	COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC Interface Reference	829
7.32.1	Detailed Description	829
7.32.2	Member Function Documentation	829
7.32.2.1	COORDINATE_CONVERT_TO_RC_DP	829
7.32.2.2	COORDINATE_CONVERT_TO_RC_SP	829
7.33	COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE Interface Reference	830
7.33.1	Detailed Description	830
7.33.2	Member Function Documentation	830
7.33.2.1	COORDINATE_DELTA_CALCULATE_DP	830
7.34	COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_CONVERT_TO_RC Interface Reference	831
7.34.1	Detailed Description	831
7.34.2	Member Function Documentation	831
7.34.2.1	COORDINATE_DERIVATIVE_CONVERT_TO_RC_DP	831

7.34.2.2	COORDINATE_DERIVATIVE_CONVERT_TO_RC_SP	831
7.35	COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY Interface Reference	832
7.35.1	Detailed Description	832
7.35.2	Member Function Documentation	832
7.35.2.1	COORDINATE_SYSTEM_DESTROY_NUMBER	832
7.35.2.2	COORDINATE_SYSTEM_DESTROY_PTR	832
7.36	COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET Interface Reference	833
7.36.1	Detailed Description	833
7.36.2	Member Function Documentation	833
7.36.2.1	COORDINATE_SYSTEM_DIMENSION_SET_NUMBER	833
7.36.2.2	COORDINATE_SYSTEM_DIMENSION_SET_PTR	833
7.37	COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET Interface Reference	834
7.37.1	Detailed Description	834
7.37.2	Member Function Documentation	834
7.37.2.1	COORDINATE_SYSTEM_FOCUS_SET_NUMBER	834
7.37.2.2	COORDINATE_SYSTEM_FOCUS_SET_PTR	834
7.38	COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIENTATION_SET Interface Reference	835
7.38.1	Detailed Description	835
7.38.2	Member Function Documentation	835
7.38.2.1	COORDINATE_SYSTEM_ORIENTATION_SET_NUMBER	835
7.38.2.2	COORDINATE_SYSTEM_ORIENTATION_SET_PTR	835
7.39	COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET Interface Reference	836
7.39.1	Detailed Description	836
7.39.2	Member Function Documentation	836
7.39.2.1	COORDINATE_SYSTEM_ORIGIN_SET_NUMBER	836
7.39.2.2	COORDINATE_SYSTEM_ORIGIN_SET_PTR	836
7.40	COORDINATE_ROUTINES::COORDINATE_SYSTEM_PTR_TYPE Struct Reference	837
7.40.1	Detailed Description	837
7.40.2	Member Data Documentation	837
7.40.2.1	PTR	837
7.41	COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_-INTERPOLATION_TYPE_SET Interface Reference	838
7.41.1	Detailed Description	838
7.41.2	Member Function Documentation	838

7.41.2.1	COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_- SET_NUMBER	838
7.41.2.2	COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_- SET_PTR	838
7.42	COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET Interface Reference	839
7.42.1	Detailed Description	839
7.42.2	Member Function Documentation	839
7.42.2.1	COORDINATE_SYSTEM_TYPE_SET_NUMBER	839
7.42.2.2	COORDINATE_SYSTEM_TYPE_SET_PTR	839
7.43	COORDINATE_ROUTINES::COORDINATE_SYSTEMS_TYPE Struct Reference	840
7.43.1	Detailed Description	840
7.43.2	Member Data Documentation	840
7.43.2.1	COORDINATE_SYSTEMS	840
7.43.2.2	NUMBER_OF_COORDINATE_SYSTEMS	840
7.44	COORDINATE_ROUTINES::D2ZX Interface Reference	841
7.44.1	Detailed Description	841
7.44.2	Member Function Documentation	841
7.44.2.1	D2ZX_DP	841
7.45	COORDINATE_ROUTINES::DXZ Interface Reference	842
7.45.1	Detailed Description	842
7.45.2	Member Function Documentation	842
7.45.2.1	DXZ_DP	842
7.46	COORDINATE_ROUTINES::DZX Interface Reference	843
7.46.1	Detailed Description	843
7.46.2	Member Function Documentation	843
7.46.2.1	DZX_DP	843
7.47	F90C::interface Interface Reference	844
7.47.1	Detailed Description	844
7.47.2	Member Function Documentation	844
7.47.2.1	CSTRINGLEN	844
7.47.2.2	PACKCHARACTERS	844
7.47.2.3	UNPACKCHARACTERS	844
7.48	FIELD_IO_ROUTINES::FIELD_IO_ELEMENTALL_INFO_SET Struct Reference	845
7.48.1	Detailed Description	845
7.48.2	Member Data Documentation	845
7.48.2.1	ELEMENTAL_INFO_SET	845
7.48.2.2	FIELDS	845

7.48.2.3	LIST_OF_GLOBAL_NUMBER	845
7.48.2.4	NUMBER_OF_ELEMENTS	846
7.49	FIELD_IO_ROUTINES::FIELD_IO_INFO_SET Struct Reference	847
7.49.1	Detailed Description	847
7.49.2	Member Data Documentation	847
7.49.2.1	COMPONENTS	847
7.49.2.2	NUMBER_OF_COMPONENTS	847
7.49.2.3	SAME_HEADER	847
7.50	FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET Struct Reference	848
7.50.1	Detailed Description	848
7.50.2	Member Data Documentation	848
7.50.2.1	FIELDS	848
7.50.2.2	LIST_OF_GLOBAL_NUMBER	848
7.50.2.3	NODAL_INFO_SET	848
7.50.2.4	NUMBER_OF_NODES	849
7.51	FIELD_IO_ROUTINES::FIELD_VARIABLE_COMPONENT_PTR_TYPE Struct Reference	850
7.51.1	Detailed Description	850
7.51.2	Member Data Documentation	850
7.51.2.1	PTR	850
7.52	FIELD_IO_ROUTINES::MESH_ELEMENTS_TYPE_PTR_TYPE Struct Reference	851
7.52.1	Detailed Description	851
7.52.2	Member Data Documentation	851
7.52.2.1	PTR	851
7.53	FIELD_ROUTINES::FIELD_COMPONENT_INTERPOLATION_SET Interface Reference	852
7.53.1	Detailed Description	852
7.53.2	Member Function Documentation	852
7.53.2.1	FIELD_COMPONENT_INTERPOLATION_SET_NUMBER	852
7.53.2.2	FIELD_COMPONENT_INTERPOLATION_SET_PTR	852
7.54	FIELD_ROUTINES::FIELD_COMPONENT_MESH_COMPONENT_SET Interface Reference	853
7.54.1	Detailed Description	853
7.54.2	Member Function Documentation	853
7.54.2.1	FIELD_COMPONENT_MESH_COMPONENT_SET_NUMBER	853
7.54.2.2	FIELD_COMPONENT_MESH_COMPONENT_SET_PTR	853
7.55	FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET Interface Reference	854
7.55.1	Detailed Description	854

7.55.2 Member Function Documentation	854
7.55.2.1 FIELD_DEPENDENT_TYPE_SET_NUMBER	854
7.55.2.2 FIELD_DEPENDENT_TYPE_SET_PTR	854
7.56 FIELD_ROUTINES::FIELD_DIMENSION_SET Interface Reference	855
7.56.1 Detailed Description	855
7.56.2 Member Function Documentation	855
7.56.2.1 FIELD_DIMENSION_SET_NUMBER	855
7.56.2.2 FIELD_DIMENSION_SET_PTR	855
7.57 FIELD_ROUTINES::FIELD_GEOMETRIC_FIELD_SET Interface Reference	856
7.57.1 Detailed Description	856
7.57.2 Member Function Documentation	856
7.57.2.1 FIELD_GEOMETRIC_FIELD_SET_NUMBER	856
7.57.2.2 FIELD_GEOMETRIC_FIELD_SET_PTR	856
7.58 FIELD_ROUTINES::FIELD_MESH_DECOMPOSITION_SET Interface Reference . . .	857
7.58.1 Detailed Description	857
7.58.2 Member Function Documentation	857
7.58.2.1 FIELD_MESH_DECOMPOSITION_SET_NUMBER	857
7.58.2.2 FIELD_MESH_DECOMPOSITION_SET_PTR	857
7.59 FIELD_ROUTINES::FIELD_NUMBER_OF_COMPONENTS_SET Interface Reference .	858
7.59.1 Detailed Description	858
7.59.2 Member Function Documentation	858
7.59.2.1 FIELD_NUMBER_OF_COMPONENTS_SET_NUMBER	858
7.59.2.2 FIELD_NUMBER_OF_COMPONENTS_SET_PTR	858
7.60 FIELD_ROUTINES::FIELD_NUMBER_OF_VARIABLES_SET Interface Reference . .	859
7.60.1 Detailed Description	859
7.60.2 Member Function Documentation	859
7.60.2.1 FIELD_NUMBER_OF_VARIABLES_SET_NUMBER	859
7.60.2.2 FIELD_NUMBER_OF_VARIABLES_SET_PTR	859
7.61 FIELD_ROUTINES::FIELD_SCALING_TYPE_SET Interface Reference	860
7.61.1 Detailed Description	860
7.61.2 Member Function Documentation	860
7.61.2.1 FIELD_SCALING_TYPE_SET_NUMBER	860
7.61.2.2 FIELD_SCALING_TYPE_SET_PTR	860
7.62 FIELD_ROUTINES::FIELD_TYPE_SET Interface Reference	861
7.62.1 Detailed Description	861
7.62.2 Member Function Documentation	861

7.62.2.1	FIELD_TYPE_SET_NUMBER	861
7.62.2.2	FIELD_TYPE_SET_PTR	861
7.63	GENERATED_MESH_ROUTINES::GENERATED_MESH_BASIS_SET Interface Reference	862
7.63.1	Detailed Description	862
7.63.2	Member Function Documentation	862
7.63.2.1	GENERATED_MESH_BASIS_SET_NUMBER	862
7.63.2.2	GENERATED_MESH_BASIS_SET_PTR	862
7.64	GENERATED_MESH_ROUTINES::GENERATED_MESH_DESTROY Interface Reference	863
7.64.1	Detailed Description	863
7.64.2	Member Function Documentation	863
7.64.2.1	GENERATED_MESH_DESTROY_NUMBER	863
7.64.2.2	GENERATED_MESH_DESTROY_PTR	863
7.65	GENERATED_MESH_ROUTINES::GENERATED_MESH_EXTENT_SET Interface Reference	864
7.65.1	Detailed Description	864
7.65.2	Member Function Documentation	864
7.65.2.1	GENERATED_MESH_EXTENT_SET_NUMBER	864
7.65.2.2	GENERATED_MESH_EXTENT_SET_PTR	864
7.66	GENERATED_MESH_ROUTINES::GENERATED_MESH_NUMBER_OF_ELEMENTS_SET Interface Reference	865
7.66.1	Detailed Description	865
7.66.2	Member Function Documentation	865
7.66.2.1	GENERATED_MESH_NUMBER_OF_ELEMENTS_SET_NUMBER	865
7.66.2.2	GENERATED_MESH_NUMBER_OF_ELEMENTS_SET_PTR	865
7.67	GENERATED_MESH_ROUTINES::GENERATED_MESH_ORIGIN_SET Interface Reference	866
7.67.1	Detailed Description	866
7.67.2	Member Function Documentation	866
7.67.2.1	GENERATED_MESH_ORIGIN_SET_NUMBER	866
7.67.2.2	GENERATED_MESH_ORIGIN_SET_PTR	866
7.68	GENERATED_MESH_ROUTINES::GENERATED_MESH_TYPE_SET Interface Reference	867
7.68.1	Detailed Description	867
7.68.2	Member Function Documentation	867
7.68.2.1	GENERATED_MESH_TYPE_SET_NUMBER	867
7.68.2.2	GENERATED_MESH_TYPE_SET_PTR	867

7.69 INPUT_OUTPUT::WRITE_STRING Interface Reference	868
7.69.1 Detailed Description	868
7.69.2 Member Function Documentation	868
7.69.2.1 WRITE_STRING_C	868
7.69.2.2 WRITE_STRING_VS	868
7.70 INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE Interface Reference	869
7.70.1 Detailed Description	869
7.70.2 Member Function Documentation	871
7.70.2.1 WRITE_STRING_FMT_TWO_VALUE_C_C	871
7.70.2.2 WRITE_STRING_FMT_TWO_VALUE_C_DP	871
7.70.2.3 WRITE_STRING_FMT_TWO_VALUE_C_INTG	871
7.70.2.4 WRITE_STRING_FMT_TWO_VALUE_C_L	871
7.70.2.5 WRITE_STRING_FMT_TWO_VALUE_C_SP	871
7.70.2.6 WRITE_STRING_FMT_TWO_VALUE_C_VS	871
7.70.2.7 WRITE_STRING_FMT_TWO_VALUE_DP_C	871
7.70.2.8 WRITE_STRING_FMT_TWO_VALUE_DP_DP	871
7.70.2.9 WRITE_STRING_FMT_TWO_VALUE_DP_INTG	871
7.70.2.10 WRITE_STRING_FMT_TWO_VALUE_DP_L	871
7.70.2.11 WRITE_STRING_FMT_TWO_VALUE_DP_SP	871
7.70.2.12 WRITE_STRING_FMT_TWO_VALUE_DP_VS	871
7.70.2.13 WRITE_STRING_FMT_TWO_VALUE_INTG_C	871
7.70.2.14 WRITE_STRING_FMT_TWO_VALUE_INTG_DP	871
7.70.2.15 WRITE_STRING_FMT_TWO_VALUE_INTG_INTG	871
7.70.2.16 WRITE_STRING_FMT_TWO_VALUE_INTG_L	871
7.70.2.17 WRITE_STRING_FMT_TWO_VALUE_INTG_SP	871
7.70.2.18 WRITE_STRING_FMT_TWO_VALUE_INTG_VS	871
7.70.2.19 WRITE_STRING_FMT_TWO_VALUE_L_C	871
7.70.2.20 WRITE_STRING_FMT_TWO_VALUE_L_DP	871
7.70.2.21 WRITE_STRING_FMT_TWO_VALUE_L_INTG	871
7.70.2.22 WRITE_STRING_FMT_TWO_VALUE_L_L	871
7.70.2.23 WRITE_STRING_FMT_TWO_VALUE_L_SP	871
7.70.2.24 WRITE_STRING_FMT_TWO_VALUE_L_VS	871
7.70.2.25 WRITE_STRING_FMT_TWO_VALUE_SP_C	871
7.70.2.26 WRITE_STRING_FMT_TWO_VALUE_SP_DP	871
7.70.2.27 WRITE_STRING_FMT_TWO_VALUE_SP_INTG	871
7.70.2.28 WRITE_STRING_FMT_TWO_VALUE_SP_L	871

7.70.2.29 WRITE_STRING_FMT_TWO_VALUE_SP_SP	871
7.70.2.30 WRITE_STRING_FMT_TWO_VALUE_SP_VS	871
7.70.2.31 WRITE_STRING_FMT_TWO_VALUE_VS_C	871
7.70.2.32 WRITE_STRING_FMT_TWO_VALUE_VS_DP	871
7.70.2.33 WRITE_STRING_FMT_TWO_VALUE_VS_INTG	871
7.70.2.34 WRITE_STRING_FMT_TWO_VALUE_VS_L	871
7.70.2.35 WRITE_STRING_FMT_TWO_VALUE_VS_SP	871
7.70.2.36 WRITE_STRING_FMT_TWO_VALUE_VS_VS	871
7.71 INPUT_OUTPUT::WRITE_STRING_FMT_VALUE Interface Reference	872
7.71.1 Detailed Description	872
7.71.2 Member Function Documentation	872
7.71.2.1 WRITE_STRING_FMT_VALUE_C	872
7.71.2.2 WRITE_STRING_FMT_VALUE_DP	873
7.71.2.3 WRITE_STRING_FMT_VALUE_INTG	873
7.71.2.4 WRITE_STRING_FMT_VALUE_L	873
7.71.2.5 WRITE_STRING_FMT_VALUE_LINTG	874
7.71.2.6 WRITE_STRING_FMT_VALUE_SP	874
7.71.2.7 WRITE_STRING_FMT_VALUE_VS	875
7.72 INPUT_OUTPUT::WRITE_STRING_IDX_VECTOR Interface Reference	876
7.72.1 Detailed Description	876
7.72.2 Member Function Documentation	876
7.72.2.1 WRITE_STRING_IDX_VECTOR_DP	876
7.72.2.2 WRITE_STRING_IDX_VECTOR_INTG	876
7.72.2.3 WRITE_STRING_IDX_VECTOR_L	876
7.72.2.4 WRITE_STRING_IDX_VECTOR_LINTG	876
7.72.2.5 WRITE_STRING_IDX_VECTOR_SP	876
7.73 INPUT_OUTPUT::WRITE_STRING_MATRIX Interface Reference	877
7.73.1 Detailed Description	877
7.73.2 Member Function Documentation	877
7.73.2.1 WRITE_STRING_MATRIX_DP	877
7.73.2.2 WRITE_STRING_MATRIX_INTG	878
7.73.2.3 WRITE_STRING_MATRIX_L	879
7.73.2.4 WRITE_STRING_MATRIX_LINTG	880
7.73.2.5 WRITE_STRING_MATRIX_SP	881
7.74 INPUT_OUTPUT::WRITE_STRING_TWO_VALUE Interface Reference	882
7.74.1 Detailed Description	883

7.74.2 Member Function Documentation	883
7.74.2.1 WRITE_STRING_TWO_VALUE_C_C	883
7.74.2.2 WRITE_STRING_TWO_VALUE_C_DP	884
7.74.2.3 WRITE_STRING_TWO_VALUE_C_INTG	884
7.74.2.4 WRITE_STRING_TWO_VALUE_C_L	885
7.74.2.5 WRITE_STRING_TWO_VALUE_C_SP	885
7.74.2.6 WRITE_STRING_TWO_VALUE_C_VS	886
7.74.2.7 WRITE_STRING_TWO_VALUE_DP_C	886
7.74.2.8 WRITE_STRING_TWO_VALUE_DP_DP	887
7.74.2.9 WRITE_STRING_TWO_VALUE_DP_INTG	887
7.74.2.10 WRITE_STRING_TWO_VALUE_DP_L	888
7.74.2.11 WRITE_STRING_TWO_VALUE_DP_SP	888
7.74.2.12 WRITE_STRING_TWO_VALUE_DP_VS	889
7.74.2.13 WRITE_STRING_TWO_VALUE_INTG_C	889
7.74.2.14 WRITE_STRING_TWO_VALUE_INTG_DP	890
7.74.2.15 WRITE_STRING_TWO_VALUE_INTG_INTG	890
7.74.2.16 WRITE_STRING_TWO_VALUE_INTG_L	891
7.74.2.17 WRITE_STRING_TWO_VALUE_INTG_SP	891
7.74.2.18 WRITE_STRING_TWO_VALUE_INTG_VS	892
7.74.2.19 WRITE_STRING_TWO_VALUE_L_C	892
7.74.2.20 WRITE_STRING_TWO_VALUE_L_DP	893
7.74.2.21 WRITE_STRING_TWO_VALUE_L_INTG	893
7.74.2.22 WRITE_STRING_TWO_VALUE_L_L	894
7.74.2.23 WRITE_STRING_TWO_VALUE_L_SP	894
7.74.2.24 WRITE_STRING_TWO_VALUE_L_VS	895
7.74.2.25 WRITE_STRING_TWO_VALUE_SP_C	895
7.74.2.26 WRITE_STRING_TWO_VALUE_SP_DP	896
7.74.2.27 WRITE_STRING_TWO_VALUE_SP_INTG	896
7.74.2.28 WRITE_STRING_TWO_VALUE_SP_L	897
7.74.2.29 WRITE_STRING_TWO_VALUE_SP_SP	897
7.74.2.30 WRITE_STRING_TWO_VALUE_SP_VS	898
7.74.2.31 WRITE_STRING_TWO_VALUE_VS_C	898
7.74.2.32 WRITE_STRING_TWO_VALUE_VS_DP	899
7.74.2.33 WRITE_STRING_TWO_VALUE_VS_INTG	899
7.74.2.34 WRITE_STRING_TWO_VALUE_VS_L	900
7.74.2.35 WRITE_STRING_TWO_VALUE_VS_SP	900

7.74.2.36 WRITE_STRING_TWO_VALUE_VS_VS	901
7.75 INPUT_OUTPUT::WRITE_STRING_VALUE Interface Reference	902
7.75.1 Detailed Description	902
7.75.2 Member Function Documentation	902
7.75.2.1 WRITE_STRING_VALUE_C	902
7.75.2.2 WRITE_STRING_VALUE_DP	902
7.75.2.3 WRITE_STRING_VALUE_INTG	903
7.75.2.4 WRITE_STRING_VALUE_L	903
7.75.2.5 WRITE_STRING_VALUE_LINTG	904
7.75.2.6 WRITE_STRING_VALUE_SP	904
7.75.2.7 WRITE_STRING_VALUE_VS	904
7.76 INPUT_OUTPUT::WRITE_STRING_VECTOR Interface Reference	905
7.76.1 Detailed Description	905
7.76.2 Member Function Documentation	905
7.76.2.1 WRITE_STRING_VECTOR_DP	905
7.76.2.2 WRITE_STRING_VECTOR_INTG	905
7.76.2.3 WRITE_STRING_VECTOR_L	905
7.76.2.4 WRITE_STRING_VECTOR_LINTG	905
7.76.2.5 WRITE_STRING_VECTOR_SP	905
7.77 ISO_VARYING_STRING::adjustr Interface Reference	906
7.77.1 Detailed Description	906
7.77.2 Member Function Documentation	906
7.77.2.1 adjustr_	906
7.78 ISO_VARYING_STRING::assignment(=) Interface Reference	907
7.78.1 Detailed Description	907
7.78.2 Member Function Documentation	907
7.78.2.1 adjustl_	907
7.78.2.2 op_assign_CH_VS	907
7.78.2.3 op_assign_VS_CH	907
7.78.2.4 op_concat_CH_VS	908
7.78.2.5 op_concat_VS_CH	908
7.78.2.6 op_concat_VS_VS	908
7.78.2.7 op_eq_CH_VS	908
7.78.2.8 op_eq_VS_CH	908
7.78.2.9 op_eq_VS_VS	908
7.78.2.10 op_ge_CH_VS	908

7.78.2.11 op_ge_VS_CH	908
7.78.2.12 op_ge_VS_VS	908
7.78.2.13 op_gt_CH_VS	909
7.78.2.14 op_gt_VS_CH	909
7.78.2.15 op_gt_VS_VS	909
7.78.2.16 op_le_CH_VS	909
7.78.2.17 op_le_VS_CH	909
7.78.2.18 op_le_VS_VS	909
7.78.2.19 op_lt_CH_VS	909
7.78.2.20 op_lt_VS_CH	909
7.78.2.21 op_lt_VS_VS	909
7.78.2.22 op_ne_CH_VS	910
7.78.2.23 op_ne_VS_CH	910
7.78.2.24 op_ne_VS_VS	910
7.79 ISO_VARYING_STRING::char Interface Reference	911
7.79.1 Detailed Description	911
7.79.2 Member Function Documentation	911
7.79.2.1 char_auto	911
7.79.2.2 char_fixed	911
7.80 ISO_VARYING_STRING::extract Interface Reference	912
7.80.1 Detailed Description	912
7.80.2 Member Function Documentation	912
7.80.2.1 extract_CH	912
7.80.2.2 extract_VS	912
7.81 ISO_VARYING_STRING::get Interface Reference	913
7.81.1 Detailed Description	913
7.81.2 Member Function Documentation	913
7.81.2.1 get_	913
7.81.2.2 get_set_CH	913
7.81.2.3 get_set_VS	913
7.81.2.4 get_unit	913
7.81.2.5 get_unit_set_CH	913
7.81.2.6 get_unit_set_VS	914
7.82 ISO_VARYING_STRING::iachar Interface Reference	915
7.82.1 Detailed Description	915
7.82.2 Member Function Documentation	915

7.82.2.1	iachar_	915
7.83	ISO_VARYING_STRING::ichar Interface Reference	916
7.83.1	Detailed Description	916
7.83.2	Member Function Documentation	916
7.83.2.1	ichar_	916
7.84	ISO_VARYING_STRING::index Interface Reference	917
7.84.1	Detailed Description	917
7.84.2	Member Function Documentation	917
7.84.2.1	index_CH_VS	917
7.84.2.2	index_VS_CH	917
7.84.2.3	index_VS_VS	917
7.85	ISO_VARYING_STRING::insert Interface Reference	918
7.85.1	Detailed Description	918
7.85.2	Member Function Documentation	918
7.85.2.1	insert_CH_CH	918
7.85.2.2	insert_CH_VS	918
7.85.2.3	insert_VS_CH	918
7.85.2.4	insert_VS_VS	918
7.86	ISO_VARYING_STRING::len Interface Reference	919
7.86.1	Detailed Description	919
7.86.2	Member Function Documentation	919
7.86.2.1	len_	919
7.87	ISO_VARYING_STRING::len_trim Interface Reference	920
7.87.1	Detailed Description	920
7.87.2	Member Function Documentation	920
7.87.2.1	len_trim_	920
7.88	ISO_VARYING_STRING::lge Interface Reference	921
7.88.1	Detailed Description	921
7.88.2	Member Function Documentation	921
7.88.2.1	lge_CH_VS	921
7.88.2.2	lge_VS_CH	921
7.88.2.3	lge_VS_VS	921
7.89	ISO_VARYING_STRING::lgt Interface Reference	922
7.89.1	Detailed Description	922
7.89.2	Member Function Documentation	922
7.89.2.1	lgt_CH_VS	922

7.89.2.2	lgt_VS_CH	922
7.89.2.3	lgt_VS_VS	922
7.90	ISO_VARYING_STRING::lle Interface Reference	923
7.90.1	Detailed Description	923
7.90.2	Member Function Documentation	923
7.90.2.1	lle_CH_VS	923
7.90.2.2	lle_VS_CH	923
7.90.2.3	lle_VS_VS	923
7.91	ISO_VARYING_STRING::llt Interface Reference	924
7.91.1	Detailed Description	924
7.91.2	Member Function Documentation	924
7.91.2.1	llt_CH_VS	924
7.91.2.2	llt_VS_CH	924
7.91.2.3	llt_VS_VS	924
7.92	ISO_VARYING_STRING::put Interface Reference	925
7.92.1	Detailed Description	925
7.92.2	Member Function Documentation	925
7.92.2.1	put_CH	925
7.92.2.2	put_unit_CH	925
7.92.2.3	put_unit_VS	925
7.92.2.4	put_VS	925
7.93	ISO_VARYING_STRING::put_line Interface Reference	926
7.93.1	Detailed Description	926
7.93.2	Member Function Documentation	926
7.93.2.1	put_line_CH	926
7.93.2.2	put_line_unit_CH	926
7.93.2.3	put_line_unit_VS	926
7.93.2.4	put_line_VS	926
7.94	ISO_VARYING_STRING::remove Interface Reference	927
7.94.1	Detailed Description	927
7.94.2	Member Function Documentation	927
7.94.2.1	remove_CH	927
7.94.2.2	remove_VS	927
7.95	ISO_VARYING_STRING::repeat Interface Reference	928
7.95.1	Detailed Description	928
7.95.2	Member Function Documentation	928

7.95.2.1	repeat_	928
7.96	ISO_VARYING_STRING::replace Interface Reference	929
7.96.1	Detailed Description	929
7.96.2	Member Function Documentation	929
7.96.2.1	replace_CH_CH_auto	929
7.96.2.2	replace_CH_CH_CH_target	929
7.96.2.3	replace_CH_CH_fixed	929
7.96.2.4	replace_CH_CH_VS_target	930
7.96.2.5	replace_CH_VS_auto	930
7.96.2.6	replace_CH_VS_CH_target	930
7.96.2.7	replace_CH_VS_fixed	930
7.96.2.8	replace_CH_VS_VS_target	930
7.96.2.9	replace_VS_CH_auto	930
7.96.2.10	replace_VS_CH_CH_target	930
7.96.2.11	replace_VS_CH_fixed	931
7.96.2.12	replace_VS_CH_VS_target	931
7.96.2.13	replace_VS_VS_auto	931
7.96.2.14	replace_VS_VS_CH_target	931
7.96.2.15	replace_VS_VS_fixed	931
7.96.2.16	replace_VS_VS_VS_target	931
7.97	ISO_VARYING_STRING::scan Interface Reference	932
7.97.1	Detailed Description	932
7.97.2	Member Function Documentation	932
7.97.2.1	scan_CH_VS	932
7.97.2.2	scan_VS_CH	932
7.97.2.3	scan_VS_VS	932
7.98	ISO_VARYING_STRING::split Interface Reference	933
7.98.1	Detailed Description	933
7.98.2	Member Function Documentation	933
7.98.2.1	split_CH	933
7.98.2.2	split_VS	933
7.99	ISO_VARYING_STRING::trim Interface Reference	934
7.99.1	Detailed Description	934
7.99.2	Member Function Documentation	934
7.99.2.1	trim_	934
7.100	ISO_VARYING_STRING::var_str Interface Reference	935

7.100.1 Detailed Description	935
7.100.2 Member Function Documentation	935
7.100.2.1 var_str_	935
7.101ISO_VARYING_STRING::VARYING_STRING Struct Reference	936
7.101.1 Detailed Description	936
7.101.2 Member Data Documentation	936
7.101.2.1 chars	936
7.102ISO_VARYING_STRING::verify Interface Reference	937
7.102.1 Detailed Description	937
7.102.2 Member Function Documentation	937
7.102.2.1 verify_CH_VS	937
7.102.2.2 verify_VS_CH	937
7.102.2.3 verify_VS_VS	937
7.103LAPACK::interface Interface Reference	938
7.103.1 Detailed Description	938
7.103.2 Member Function Documentation	938
7.103.2.1 DGESV	938
7.104LISTS::LIST_DETACH_AND_DESTROY Interface Reference	939
7.104.1 Detailed Description	939
7.104.2 Member Function Documentation	939
7.104.2.1 LIST_DETACH_AND_DESTROY_DP	939
7.104.2.2 LIST_DETACH_AND_DESTROY_INTG	940
7.104.2.3 LIST_DETACH_AND_DESTROY_SP	940
7.105LISTS::LIST_ITEM_ADD Interface Reference	941
7.105.1 Detailed Description	941
7.105.2 Member Function Documentation	941
7.105.2.1 LIST_ITEM_ADD_DP1	941
7.105.2.2 LIST_ITEM_ADD_INTG1	941
7.105.2.3 LIST_ITEM_ADD_SP1	942
7.106LISTS::LIST_ITEM_IN_LIST Interface Reference	943
7.106.1 Detailed Description	943
7.106.2 Member Function Documentation	943
7.106.2.1 LIST_ITEM_IN_LIST_DP1	943
7.106.2.2 LIST_ITEM_IN_LIST_INTG1	943
7.106.2.3 LIST_ITEM_IN_LIST_SP1	944
7.107LISTS::LIST_PTR_TYPE Struct Reference	945

7.107.1 Detailed Description	945
7.107.2 Member Data Documentation	945
7.107.2.1 PTR	945
7.108LISTS::LIST_SEARCH Interface Reference	946
7.108.1 Detailed Description	946
7.108.2 Member Function Documentation	946
7.108.2.1 LIST_SEARCH_DP_ARRAY	946
7.108.2.2 LIST_SEARCH_INTG_ARRAY	946
7.108.2.3 LIST_SEARCH_SP_ARRAY	947
7.109LISTS::LIST_SEARCH_LINEAR Interface Reference	948
7.109.1 Detailed Description	948
7.109.2 Member Function Documentation	948
7.109.2.1 LIST_SEARCH_LINEAR_DP_ARRAY	948
7.109.2.2 LIST_SEARCH_LINEAR_INTG_ARRAY	948
7.109.2.3 LIST_SEARCH_LINEAR_SP_ARRAY	949
7.110LISTS::LIST_SORT Interface Reference	950
7.110.1 Detailed Description	950
7.110.2 Member Function Documentation	950
7.110.2.1 LIST_SORT_DP_ARRAY	950
7.110.2.2 LIST_SORT_INTG_ARRAY	950
7.110.2.3 LIST_SORT_SP_ARRAY	950
7.111LISTS::LIST_SORT_BUBBLE Interface Reference	952
7.111.1 Detailed Description	952
7.111.2 Member Function Documentation	952
7.111.2.1 LIST_SORT_BUBBLE_DP_ARRAY	952
7.111.2.2 LIST_SORT_BUBBLE_INTG_ARRAY	952
7.111.2.3 LIST_SORT_BUBBLE_SP_ARRAY	952
7.112LISTS::LIST_SORT_HEAP Interface Reference	954
7.112.1 Detailed Description	954
7.112.2 Member Function Documentation	954
7.112.2.1 LIST_SORT_HEAP_DP_ARRAY	954
7.112.2.2 LIST_SORT_HEAP_INTG_ARRAY	954
7.112.2.3 LIST_SORT_HEAP_SP_ARRAY	954
7.113LISTS::LIST_SORT_SHELL Interface Reference	956
7.113.1 Detailed Description	956
7.113.2 Member Function Documentation	956

7.113.2.1 LIST_SORT_SHELL_DP_ARRAY	956
7.113.2.2 LIST_SORT_SHELL_INTG_ARRAY	956
7.113.2.3 LIST_SORT_SHELL_SP_ARRAY	956
7.114 LISTS::LIST_TYPE Struct Reference	957
7.114.1 Detailed Description	957
7.114.2 Member Data Documentation	957
7.114.2.1 DATA_TYPE	957
7.114.2.2 INITIAL_SIZE	958
7.114.2.3 LIST_DP	958
7.114.2.4 LIST_FINISHED	958
7.114.2.5 LIST_INTG	958
7.114.2.6 LIST_SP	958
7.114.2.7 NUMBER_IN_LIST	958
7.114.2.8 SIZE	958
7.114.2.9 SORT_METHOD	958
7.114.2.10 SORT_ORDER	959
7.115 MATHS::CROSS_PRODUCT Interface Reference	960
7.115.1 Detailed Description	960
7.115.2 Member Function Documentation	960
7.115.2.1 CROSS_PRODUCT_DP	960
7.115.2.2 CROSS_PRODUCT_INTG	960
7.115.2.3 CROSS_PRODUCT_SP	960
7.116 MATHS::D_CROSS_PRODUCT Interface Reference	961
7.116.1 Detailed Description	961
7.116.2 Member Function Documentation	961
7.116.2.1 D_CROSS_PRODUCT_DP	961
7.116.2.2 D_CROSS_PRODUCT_INTG	961
7.116.2.3 D_CROSS_PRODUCT_SP	961
7.117 MATHS::DETERMINANT Interface Reference	962
7.117.1 Detailed Description	962
7.117.2 Member Function Documentation	962
7.117.2.1 DETERMINANT_FULL_DP	962
7.117.2.2 DETERMINANT_FULL_INTG	962
7.117.2.3 DETERMINANT_FULL_SP	962
7.118 MATHS::EDP Interface Reference	963
7.118.1 Detailed Description	963

7.118.2 Member Function Documentation	963
7.118.2.1 EDP_DP	963
7.118.2.2 EDP_SP	963
7.119MATHS::EIGENVALUE Interface Reference	964
7.119.1 Detailed Description	964
7.119.2 Member Function Documentation	964
7.119.2.1 EIGENVALUE_FULL_DP	964
7.119.2.2 EIGENVALUE_FULL_SP	964
7.120MATHS::EIGENVECTOR Interface Reference	965
7.120.1 Detailed Description	965
7.120.2 Member Function Documentation	965
7.120.2.1 EIGENVECTOR_FULL_DP	965
7.120.2.2 EIGENVECTOR_FULL_SP	965
7.121MATHS::I0 Interface Reference	966
7.121.1 Detailed Description	966
7.121.2 Member Function Documentation	966
7.121.2.1 I0_DP	966
7.121.2.2 I0_SP	966
7.122MATHS::I1 Interface Reference	967
7.122.1 Detailed Description	967
7.122.2 Member Function Documentation	967
7.122.2.1 I1_DP	967
7.122.2.2 I1_SP	967
7.123MATHS::INVERT Interface Reference	968
7.123.1 Detailed Description	968
7.123.2 Member Function Documentation	968
7.123.2.1 INVERT_FULL_DP	968
7.123.2.2 INVERT_FULL_SP	968
7.124MATHS::K0 Interface Reference	969
7.124.1 Detailed Description	969
7.124.2 Member Function Documentation	969
7.124.2.1 K0_DP	969
7.124.2.2 K0_SP	969
7.125MATHS::K1 Interface Reference	970
7.125.1 Detailed Description	970
7.125.2 Member Function Documentation	970

7.125.2.1 K1_DP	970
7.125.2.2 K1_SP	970
7.126MATHS::L2NORM Interface Reference	971
7.126.1 Detailed Description	971
7.126.2 Member Function Documentation	971
7.126.2.1 L2NORM_DP	971
7.126.2.2 L2NORM_SP	971
7.127MATHS::MATRIX_PRODUCT Interface Reference	972
7.127.1 Detailed Description	972
7.127.2 Member Function Documentation	972
7.127.2.1 MATRIX_PRODUCT_DP	972
7.127.2.2 MATRIX_PRODUCT_SP	972
7.128MATHS::MATRIX_TRANSPOSE Interface Reference	973
7.128.1 Detailed Description	973
7.128.2 Member Function Documentation	973
7.128.2.1 MATRIX_TRANSPOSE_DP	973
7.128.2.2 MATRIX_TRANSPOSE_SP	973
7.129MATHS::NORMALISE Interface Reference	974
7.129.1 Detailed Description	974
7.129.2 Member Function Documentation	974
7.129.2.1 NORMALISE_DP	974
7.129.2.2 NORMALISE_SP	974
7.130MATHS::SOLVE_SMALL_LINEAR_SYSTEM Interface Reference	975
7.130.1 Detailed Description	975
7.130.2 Member Function Documentation	975
7.130.2.1 SOLVE_SMALL_LINEAR_SYSTEM_DP	975
7.130.2.2 SOLVE_SMALL_LINEAR_SYSTEM_SP	975
7.131MATRIX_VECTOR::MATRIX_ALL_VALUES_SET Interface Reference	976
7.131.1 Detailed Description	976
7.131.2 Member Function Documentation	976
7.131.2.1 MATRIX_ALL_VALUES_SET_DP	976
7.131.2.2 MATRIX_ALL_VALUES_SET_INTG	976
7.131.2.3 MATRIX_ALL_VALUES_SET_L	977
7.131.2.4 MATRIX_ALL_VALUES_SET_SP	977
7.132MATRIX_VECTOR::MATRIX_DATA_GET Interface Reference	978
7.132.1 Detailed Description	978

7.132.2 Member Function Documentation	978
7.132.2.1 MATRIX_DATA_GET_DP	978
7.132.2.2 MATRIX_DATA_GET_INTG	978
7.132.2.3 MATRIX_DATA_GET_L	979
7.132.2.4 MATRIX_DATA_GET_SP	979
7.133 MATRIX_VECTOR::MATRIX_VALUES_ADD Interface Reference	980
7.133.1 Detailed Description	980
7.133.2 Member Function Documentation	980
7.133.2.1 MATRIX_VALUES_ADD_DP	980
7.133.2.2 MATRIX_VALUES_ADD_DP1	981
7.133.2.3 MATRIX_VALUES_ADD_DP2	981
7.133.2.4 MATRIX_VALUES_ADD_INTG	981
7.133.2.5 MATRIX_VALUES_ADD_INTG1	982
7.133.2.6 MATRIX_VALUES_ADD_INTG2	982
7.133.2.7 MATRIX_VALUES_ADD_L	983
7.133.2.8 MATRIX_VALUES_ADD_L1	983
7.133.2.9 MATRIX_VALUES_ADD_L2	983
7.133.2.10 MATRIX_VALUES_ADD_SP	984
7.133.2.11 MATRIX_VALUES_ADD_SP1	984
7.133.2.12 MATRIX_VALUES_ADD_SP2	984
7.134 MATRIX_VECTOR::MATRIX_VALUES_GET Interface Reference	985
7.134.1 Detailed Description	985
7.134.2 Member Function Documentation	985
7.134.2.1 MATRIX_VALUES_GET_DP	985
7.134.2.2 MATRIX_VALUES_GET_DP1	986
7.134.2.3 MATRIX_VALUES_GET_DP2	986
7.134.2.4 MATRIX_VALUES_GET_INTG	986
7.134.2.5 MATRIX_VALUES_GET_INTG1	987
7.134.2.6 MATRIX_VALUES_GET_INTG2	987
7.134.2.7 MATRIX_VALUES_GET_L	988
7.134.2.8 MATRIX_VALUES_GET_L1	988
7.134.2.9 MATRIX_VALUES_GET_L2	988
7.134.2.10 MATRIX_VALUES_GET_SP	989
7.134.2.11 MATRIX_VALUES_GET_SP1	989
7.134.2.12 MATRIX_VALUES_GET_SP2	989
7.135 MATRIX_VECTOR::MATRIX_VALUES_SET Interface Reference	990

7.135.1 Detailed Description	990
7.135.2 Member Function Documentation	990
7.135.2.1 MATRIX_VALUES_SET_DP	990
7.135.2.2 MATRIX_VALUES_SET_DP1	991
7.135.2.3 MATRIX_VALUES_SET_DP2	991
7.135.2.4 MATRIX_VALUES_SET_INTG	991
7.135.2.5 MATRIX_VALUES_SET_INTG1	992
7.135.2.6 MATRIX_VALUES_SET_INTG2	992
7.135.2.7 MATRIX_VALUES_SET_L	992
7.135.2.8 MATRIX_VALUES_SET_L1	993
7.135.2.9 MATRIX_VALUES_SET_L2	993
7.135.2.10 MATRIX_VALUES_SET_SP	994
7.135.2.11 MATRIX_VALUES_SET_SP1	994
7.135.2.12 MATRIX_VALUES_SET_SP2	994
7.136 MATRIX_VECTOR::VECTOR_ALL_VALUES_SET Interface Reference	995
7.136.1 Detailed Description	995
7.136.2 Member Function Documentation	995
7.136.2.1 VECTOR_ALL_VALUES_SET_DP	995
7.136.2.2 VECTOR_ALL_VALUES_SET_INTG	995
7.136.2.3 VECTOR_ALL_VALUES_SET_L	996
7.136.2.4 VECTOR_ALL_VALUES_SET_SP	996
7.137 MATRIX_VECTOR::VECTOR_DATA_GET Interface Reference	997
7.137.1 Detailed Description	997
7.137.2 Member Function Documentation	997
7.137.2.1 VECTOR_DATA_GET_DP	997
7.137.2.2 VECTOR_DATA_GET_INTG	997
7.137.2.3 VECTOR_DATA_GET_L	998
7.137.2.4 VECTOR_DATA_GET_SP	998
7.138 MATRIX_VECTOR::VECTOR_VALUES_GET Interface Reference	999
7.138.1 Detailed Description	999
7.138.2 Member Function Documentation	999
7.138.2.1 VECTOR_VALUES_GET_DP	999
7.138.2.2 VECTOR_VALUES_GET_DP1	999
7.138.2.3 VECTOR_VALUES_GET_INTG	1000
7.138.2.4 VECTOR_VALUES_GET_INTG1	1000
7.138.2.5 VECTOR_VALUES_GET_L	1000

7.138.2.6 VECTOR_VALUES_GET_L1	1001
7.138.2.7 VECTOR_VALUES_GET_SP	1001
7.138.2.8 VECTOR_VALUES_GET_SP1	1001
7.139 MATRIX_VECTOR::VECTOR_VALUES_SET Interface Reference	1002
7.139.1 Detailed Description	1002
7.139.2 Member Function Documentation	1002
7.139.2.1 VECTOR_VALUES_SET_DP	1002
7.139.2.2 VECTOR_VALUES_SET_DP1	1002
7.139.2.3 VECTOR_VALUES_SET_INTG	1003
7.139.2.4 VECTOR_VALUES_SET_INTG1	1003
7.139.2.5 VECTOR_VALUES_SET_L	1003
7.139.2.6 VECTOR_VALUES_SET_L1	1004
7.139.2.7 VECTOR_VALUES_SET_SP	1004
7.139.2.8 VECTOR_VALUES_SET_SP1	1004
7.140 MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET Interface Reference	1005
7.140.1 Detailed Description	1005
7.140.2 Member Function Documentation	1005
7.140.2.1 MESH_NUMBER_OF_COMPONENTS_SET_NUMBER	1005
7.140.2.2 MESH_NUMBER_OF_COMPONENTS_SET_PTR	1005
7.141 MESH_ROUTINES::MESH_NUMBER_OF_ELEMENTS_SET Interface Reference	1006
7.141.1 Detailed Description	1006
7.141.2 Member Function Documentation	1006
7.141.2.1 MESH_NUMBER_OF_ELEMENTS_SET_NUMBER	1006
7.141.2.2 MESH_NUMBER_OF_ELEMENTS_SET_PTR	1006
7.142 PROBLEM_ROUTINES::PROBLEM_DESTROY Interface Reference	1007
7.142.1 Detailed Description	1007
7.142.2 Member Function Documentation	1007
7.142.2.1 PROBLEM_DESTROY_NUMBER	1007
7.142.2.2 PROBLEM_DESTROY_PTR	1007
7.143 PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_GET Interface Reference	1008
7.143.1 Detailed Description	1008
7.143.2 Member Function Documentation	1008
7.143.2.1 PROBLEM_SPECIFICATION_GET_NUMBER	1008
7.143.2.2 PROBLEM_SPECIFICATION_GET_PTR	1008
7.144 PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET Interface Reference	1010
7.144.1 Detailed Description	1010

7.144.2 Member Function Documentation	1010
7.144.2.1 PROBLEM_SPECIFICATION_SET_NUMBER	1010
7.144.2.2 PROBLEM_SPECIFICATION_SET_PTR	1010
7.145REGION_ROUTINES::REGION_COORDINATE_SYSTEM_SET Interface Reference . .	1012
7.145.1 Detailed Description	1012
7.145.2 Member Function Documentation	1012
7.145.2.1 REGION_COORDINATE_SYSTEM_SET_NUMBER	1012
7.145.2.2 REGION_COORDINATE_SYSTEM_SET_PTR	1012
7.146REGION_ROUTINES::REGION_LABEL_SET Interface Reference	1013
7.146.1 Detailed Description	1013
7.146.2 Member Function Documentation	1013
7.146.2.1 REGION_LABEL_SET_NUMBER	1013
7.146.2.2 REGION_LABEL_SET_PTR	1013
7.147SORTING::BUBBLE_SORT Interface Reference	1014
7.147.1 Detailed Description	1014
7.147.2 Member Function Documentation	1014
7.147.2.1 BUBBLE_SORT_DP	1014
7.147.2.2 BUBBLE_SORT_INTG	1014
7.147.2.3 BUBBLE_SORT_SP	1014
7.148SORTING::HEAP_SORT Interface Reference	1015
7.148.1 Detailed Description	1015
7.148.2 Member Function Documentation	1015
7.148.2.1 HEAP_SORT_DP	1015
7.148.2.2 HEAP_SORT_INTG	1015
7.148.2.3 HEAP_SORT_SP	1015
7.149SORTING::SHELL_SORT Interface Reference	1016
7.149.1 Detailed Description	1016
7.149.2 Member Function Documentation	1016
7.149.2.1 SHELL_SORT_DP	1016
7.149.2.2 SHELL_SORT_INTG	1016
7.149.2.3 SHELL_SORT_SP	1016
7.150STRINGS::CHARACTER_TO_LOWERCASE Interface Reference	1017
7.150.1 Detailed Description	1017
7.150.2 Member Function Documentation	1017
7.150.2.1 CHARACTER_TO_LOWERCASE_C	1017
7.150.2.2 CHARACTER_TO_LOWERCASE_VS	1017

7.151STRINGS::CHARACTER_TO_UPPERCASE Interface Reference	1018
7.151.1 Detailed Description	1018
7.151.2 Member Function Documentation	1018
7.151.2.1 CHARACTER_TO_UPPERCASE_C	1018
7.151.2.2 CHARACTER_TO_UPPERCASE_VS	1018
7.152STRINGS::IS_ABBREVIATION Interface Reference	1019
7.152.1 Detailed Description	1019
7.152.2 Member Function Documentation	1019
7.152.2.1 IS_ABBREVIATION_C_C	1019
7.152.2.2 IS_ABBREVIATION_C_VS	1019
7.152.2.3 IS_ABBREVIATION_VS_C	1019
7.152.2.4 IS_ABBREVIATION_VS_VS	1020
7.153STRINGS::LIST_TO_CHARACTER Interface Reference	1021
7.153.1 Detailed Description	1021
7.153.2 Member Function Documentation	1021
7.153.2.1 LIST_TO_CHARACTER_C	1021
7.153.2.2 LIST_TO_CHARACTER_DP	1022
7.153.2.3 LIST_TO_CHARACTER_INTG	1022
7.153.2.4 LIST_TO_CHARACTER_L	1022
7.153.2.5 LIST_TO_CHARACTER_LINTG	1023
7.153.2.6 LIST_TO_CHARACTER_SP	1023
7.154STRINGS::NUMBER_TO_CHARACTER Interface Reference	1024
7.154.1 Detailed Description	1024
7.154.2 Member Function Documentation	1024
7.154.2.1 NUMBER_TO_CHARACTER_DP	1024
7.154.2.2 NUMBER_TO_CHARACTER_INTG	1024
7.154.2.3 NUMBER_TO_CHARACTER_LINTG	1025
7.154.2.4 NUMBER_TO_CHARACTER_SP	1025
7.155STRINGS::NUMBER_TO_VSTRING Interface Reference	1026
7.155.1 Detailed Description	1026
7.155.2 Member Function Documentation	1026
7.155.2.1 NUMBER_TO_VSTRING_DP	1026
7.155.2.2 NUMBER_TO_VSTRING_INTG	1026
7.155.2.3 NUMBER_TO_VSTRING_LINTG	1027
7.155.2.4 NUMBER_TO_VSTRING_SP	1027
7.156STRINGS::STRING_TO_DOUBLE Interface Reference	1028

7.156.1 Detailed Description	1028
7.156.2 Member Function Documentation	1028
7.156.2.1 STRING_TO_DOUBLE_C	1028
7.156.2.2 STRING_TO_DOUBLE_VS	1028
7.157 STRINGS::STRING_TO_INTEGER Interface Reference	1029
7.157.1 Detailed Description	1029
7.157.2 Member Function Documentation	1029
7.157.2.1 STRING_TO_INTEGER_C	1029
7.157.2.2 STRING_TO_INTEGER_VS	1029
7.158 STRINGS::STRING_TO_LOGICAL Interface Reference	1030
7.158.1 Detailed Description	1030
7.158.2 Member Function Documentation	1030
7.158.2.1 STRING_TO_LOGICAL_C	1030
7.158.2.2 STRING_TO_LOGICAL_VS	1030
7.159 STRINGS::STRING_TO_LONG_INTEGER Interface Reference	1031
7.159.1 Detailed Description	1031
7.159.2 Member Function Documentation	1031
7.159.2.1 STRING_TO_LONG_INTEGER_C	1031
7.159.2.2 STRING_TO_LONG_INTEGER_VS	1031
7.160 STRINGS::STRING_TO_SINGLE Interface Reference	1032
7.160.1 Detailed Description	1032
7.160.2 Member Function Documentation	1032
7.160.2.1 STRING_TO_SINGLE_C	1032
7.160.2.2 STRING_TO_SINGLE_VS	1032
7.161 VSTRINGS::VSTRING_TO_LOWERCASE Interface Reference	1033
7.161.1 Detailed Description	1033
7.161.2 Member Function Documentation	1033
7.161.2.1 VSTRING_TO_LOWERCASE_C	1033
7.161.2.2 VSTRING_TO_LOWERCASE_VS	1033
7.162 VSTRINGS::VSTRING_TO_UPPERCASE Interface Reference	1034
7.162.1 Detailed Description	1034
7.162.2 Member Function Documentation	1034
7.162.2.1 VSTRING_TO_UPPERCASE_C	1034
7.162.2.2 VSTRING_TO_UPPERCASE_VS	1034
7.163 TIMER::interface Interface Reference	1035
7.163.1 Detailed Description	1035

7.163.2 Member Function Documentation	1035
7.163.2.1 CPUTIMER	1035
7.164 TREES::TREE_NODE_TYPE Struct Reference	1036
7.164.1 Detailed Description	1036
7.164.2 Member Data Documentation	1036
7.164.2.1 COLOUR	1036
7.164.2.2 KEY	1036
7.164.2.3 LEFT	1036
7.164.2.4 PARENT	1037
7.164.2.5 RIGHT	1037
7.164.2.6 VALUE	1037
7.165 TREES::TREE_TYPE Struct Reference	1038
7.165.1 Detailed Description	1038
7.165.2 Member Data Documentation	1038
7.165.2.1 INSERT_TYPE	1038
7.165.2.2 NIL	1038
7.165.2.3 NUMBER_IN_TREE	1038
7.165.2.4 ROOT	1038
7.165.2.5 TREE_FINISHED	1039
7.166 TYPES::BASIS_FUNCTIONS_TYPE Struct Reference	1040
7.166.1 Detailed Description	1040
7.166.2 Member Data Documentation	1040
7.166.2.1 BASES	1040
7.166.2.2 NUMBER_BASIS_FUNCTIONS	1040
7.167 TYPES::BASIS_PTR_TYPE Struct Reference	1041
7.167.1 Detailed Description	1041
7.167.2 Member Data Documentation	1041
7.167.2.1 PTR	1041
7.168 TYPES::BASIS_TYPE Struct Reference	1042
7.168.1 Detailed Description	1045
7.168.2 Member Data Documentation	1045
7.168.2.1 BASIS_FINISHED	1045
7.168.2.2 COLLAPSED_XI	1045
7.168.2.3 DEGENERATE	1045
7.168.2.4 DERIVATIVE_NUMBERS_IN_LOCAL_LINE	1046
7.168.2.5 DERIVATIVE_ORDER_INDEX	1046

7.168.2.6 DERIVATIVE_ORDER_INDEX_INV	1046
7.168.2.7 ELEMENT_PARAMETER_INDEX	1046
7.168.2.8 FACE_BASES	1046
7.168.2.9 FAMILY_NUMBER	1047
7.168.2.10 GLOBAL_NUMBER	1047
7.168.2.11 HERMITE	1047
7.168.2.12 INTERPOLATION_ORDER	1047
7.168.2.13 INTERPOLATION_TYPE	1047
7.168.2.14 INTERPOLATION_XI	1047
7.168.2.15 LINE_BASES	1048
7.168.2.16 LOCAL_LINE_XI_DIRECTION	1048
7.168.2.17 MAXIMUM_NUMBER_OF_DERIVATIVES	1048
7.168.2.18 NODE_AT_COLLAPSE	1048
7.168.2.19 NODE_NUMBERS_IN_LOCAL_LINE	1048
7.168.2.20 NODE_POSITION_INDEX	1048
7.168.2.21 NODE_POSITION_INDEX_INV	1049
7.168.2.22 NUMBER_OF_COLLAPSED_XI	1049
7.168.2.23 NUMBER_OF_DERIVATIVES	1049
7.168.2.24 NUMBER_OF_ELEMENT_PARAMETERS	1049
7.168.2.25 NUMBER_OF_LOCAL_LINES	1049
7.168.2.26 NUMBER_OF_NODES	1049
7.168.2.27 NUMBER_OF_NODES_IN_LOCAL_LINE	1049
7.168.2.28 NUMBER_OF_NODES_XI	1050
7.168.2.29 NUMBER_OF_PARTIAL_DERIVATIVES	1050
7.168.2.30 NUMBER_OF_SUB_BASES	1050
7.168.2.31 NUMBER_OF_XI	1050
7.168.2.32 NUMBER_OF_XI_COORDINATES	1050
7.168.2.33 PARENT_BASIS	1050
7.168.2.34 PARTIAL_DERIVATIVE_INDEX	1050
7.168.2.35 QUADRATURE	1051
7.168.2.36 SUB_BASES	1051
7.168.2.37 TYPE	1051
7.168.2.38 USER_NUMBER	1051
7.169 TYPES::COORDINATE_SYSTEM_TYPE Struct Reference	1052
7.169.1 Detailed Description	1052
7.169.2 Member Data Documentation	1053

7.169.2.1 COORDINATE_SYSTEM_FINISHED	1053
7.169.2.2 FOCUS	1053
7.169.2.3 NUMBER_OF_DIMENSIONS	1053
7.169.2.4 ORIENTATION	1053
7.169.2.5 ORIGIN	1053
7.169.2.6 RADIAL_INTERPOLATION_TYPE	1053
7.169.2.7 TYPE	1053
7.169.2.8 USER_NUMBER	1054
7.170TYPES::DECOMPOSITION_ELEMENT_TYPE Struct Reference	1055
7.170.1 Detailed Description	1055
7.170.2 Member Data Documentation	1055
7.170.2.1 ADJACENT_ELEMENTS	1055
7.170.2.2 ELEMENT_LINES	1056
7.170.2.3 GLOBAL_NUMBER	1056
7.170.2.4 LOCAL_NUMBER	1056
7.170.2.5 NUMBER_OF_ADJACENT_ELEMENTS	1056
7.170.2.6 USER_NUMBER	1056
7.171TYPES::DECOMPOSITION_ELEMENTS_TYPE Struct Reference	1057
7.171.1 Detailed Description	1057
7.171.2 Member Data Documentation	1057
7.171.2.1 DECOMPOSITION	1057
7.171.2.2 ELEMENTS	1057
7.171.2.3 TOTAL_NUMBER_OF_ELEMENTS	1058
7.172TYPES::DECOMPOSITION_LINE_TYPE Struct Reference	1059
7.172.1 Detailed Description	1059
7.172.2 Member Data Documentation	1059
7.172.2.1 ADJACENT_LINES	1059
7.172.2.2 ELEMENT_LINES	1059
7.172.2.3 NUMBER	1060
7.172.2.4 NUMBER_OF_SURROUNDING_ELEMENTS	1060
7.172.2.5 SURROUNDING_ELEMENTS	1060
7.172.2.6 XI_DIRECTION	1060
7.173TYPES::DECOMPOSITION_LINES_TYPE Struct Reference	1061
7.173.1 Detailed Description	1061
7.173.2 Member Data Documentation	1061
7.173.2.1 DECOMPOSITION	1061

7.173.2.2 LINES	1061
7.173.2.3 NUMBER_OF_LINES	1061
7.174TYPES::DECOMPOSITION_PTR_TYPE Struct Reference	1062
7.174.1 Detailed Description	1062
7.174.2 Member Data Documentation	1062
7.174.2.1 PTR	1062
7.175TYPES::DECOMPOSITION_TOPOLOGY_TYPE Struct Reference	1063
7.175.1 Detailed Description	1063
7.175.2 Member Data Documentation	1063
7.175.2.1 DECOMPOSITION	1063
7.175.2.2 ELEMENTS	1063
7.175.2.3 LINES	1063
7.176TYPES::DECOMPOSITION_TYPE Struct Reference	1064
7.176.1 Detailed Description	1065
7.176.2 Member Data Documentation	1065
7.176.2.1 DECOMPOSITION_FINISHED	1065
7.176.2.2 DECOMPOSITION_TYPE	1065
7.176.2.3 DECOMPOSITIONS	1065
7.176.2.4 DOMAIN	1065
7.176.2.5 ELEMENT_DOMAIN	1065
7.176.2.6 GLOBAL_NUMBER	1066
7.176.2.7 MESH	1066
7.176.2.8 MESH_COMPONENT_NUMBER	1066
7.176.2.9 NUMBER_OF_DOMAINS	1066
7.176.2.10 NUMBER_OF_EDGES_CUT	1066
7.176.2.11 TOPOLOGY	1066
7.176.2.12 USER_NUMBER	1066
7.177TYPES::DECOMPOSITIONS_TYPE Struct Reference	1067
7.177.1 Detailed Description	1067
7.177.2 Member Data Documentation	1067
7.177.2.1 DECOMPOSITIONS	1067
7.177.2.2 MESH	1067
7.177.2.3 NUMBER_OF_DECOMPOSITIONS	1067
7.178TYPES::DISTRIBUTED_MATRIX_CMISS_TYPE Struct Reference	1068
7.178.1 Detailed Description	1068
7.178.2 Member Data Documentation	1068

7.178.2.1 BASE_TAG_NUMBER	1068
7.178.2.2 DISTRIBUTED_MATRIX	1068
7.178.2.3 MATRIX	1068
7.179TYPES::DISTRIBUTED_MATRIX_PETSC_TYPE Struct Reference	1069
7.179.1 Detailed Description	1070
7.179.2 Member Data Documentation	1070
7.179.2.1 COLUMN_INDICES	1070
7.179.2.2 DATA_DP	1070
7.179.2.3 DATA_SIZE	1070
7.179.2.4 DIAGONAL_NUMBER_NON_ZEROS	1070
7.179.2.5 DISTRIBUTED_MATRIX	1070
7.179.2.6 GLOBAL_M	1071
7.179.2.7 GLOBAL_N	1071
7.179.2.8 GLOBAL_ROW_NUMBERS	1071
7.179.2.9 ISLTGMAPPING	1071
7.179.2.10M	1071
7.179.2.11IMATRIX	1071
7.179.2.12MAXIMUM_COLUMN_INDICES_PER_ROW	1071
7.179.2.13N	1071
7.179.2.14NUMBER_NON_ZEROS	1072
7.179.2.15OFFDIAGONAL_NUMBER_NON_ZEROS	1072
7.179.2.16ROW_INDICES	1072
7.179.2.17STORAGE_TYPE	1072
7.180TYPES::DISTRIBUTED_MATRIX_TYPE Struct Reference	1073
7.180.1 Detailed Description	1073
7.180.2 Member Data Documentation	1073
7.180.2.1 CMIS	1073
7.180.2.2 COLUMN_DOMAIN_MAPPING	1074
7.180.2.3 DATA_TYPE	1074
7.180.2.4 GHOSTING_TYPE	1074
7.180.2.5 LIBRARY_TYPE	1074
7.180.2.6 MATRIX_FINISHED	1074
7.180.2.7 PETSC	1074
7.180.2.8 ROW_DOMAIN_MAPPING	1075
7.181TYPES::DISTRIBUTED_VECTOR_CMIS_TYPE Struct Reference	1076
7.181.1 Detailed Description	1076

7.181.2 Member Data Documentation	1077
7.181.2.1 BASE_TAG_NUMBER	1077
7.181.2.2 DATA_DP	1077
7.181.2.3 DATA_INTG	1077
7.181.2.4 DATA_L	1077
7.181.2.5 DATA_SIZE	1077
7.181.2.6 DATA_SP	1077
7.181.2.7 DISTRIBUTED_VECTOR	1077
7.181.2.8 N	1078
7.181.2.9 TRANSFERS	1078
7.182TYPES::DISTRIBUTED_VECTOR_PETSC_TYPE Struct Reference	1079
7.182.1 Detailed Description	1079
7.182.2 Member Data Documentation	1079
7.182.2.1 DATA_SIZE	1079
7.182.2.2 DISTRIBUTED_VECTOR	1079
7.182.2.3 GLOBAL_N	1080
7.182.2.4 GLOBAL_NUMBERS	1080
7.182.2.5 ISLTGMAPPING	1080
7.182.2.6 N	1080
7.182.2.7 VECTOR	1080
7.183TYPES::DISTRIBUTED_VECTOR_TRANSFER_TYPE Struct Reference	1081
7.183.1 Detailed Description	1082
7.183.2 Member Data Documentation	1082
7.183.2.1 CMISS_VECTOR	1082
7.183.2.2 DATA_TYPE	1082
7.183.2.3 MPI_RECEIVE_REQUEST	1082
7.183.2.4 MPI_SEND_REQUEST	1082
7.183.2.5 RECEIVE_BUFFER_DP	1083
7.183.2.6 RECEIVE_BUFFER_INTG	1083
7.183.2.7 RECEIVE_BUFFER_L	1083
7.183.2.8 RECEIVE_BUFFER_SIZE	1083
7.183.2.9 RECEIVE_BUFFER_SP	1083
7.183.2.10RECEIVE_TAG_NUMBER	1083
7.183.2.11SEND_BUFFER_DP	1084
7.183.2.12SEND_BUFFER_INTG	1084
7.183.2.13SEND_BUFFER_L	1084

7.183.2.14SEND_BUFFER_SIZE	1084
7.183.2.15SEND_BUFFER_SP	1084
7.183.2.16SEND_TAG_NUMBER	1084
7.184TYPES::DISTRIBUTED_VECTOR_TYPE Struct Reference	1085
7.184.1 Detailed Description	1085
7.184.2 Member Data Documentation	1085
7.184.2.1 CMISS	1085
7.184.2.2 DATA_TYPE	1085
7.184.2.3 DOMAIN_MAPPING	1086
7.184.2.4 GHOSTING_TYPE	1086
7.184.2.5 LIBRARY_TYPE	1086
7.184.2.6 PETSC	1086
7.184.2.7 VECTOR_FINISHED	1086
7.185TYPES::DOMAIN_ADJACENT_DOMAIN_TYPE Struct Reference	1087
7.185.1 Detailed Description	1087
7.185.2 Member Data Documentation	1087
7.185.2.1 DOMAIN_NUMBER	1087
7.185.2.2 LOCAL_GHOST_RECEIVE_INDICES	1087
7.185.2.3 LOCAL_GHOST_SEND_INDICES	1088
7.185.2.4 NUMBER_OF_RECEIVE_GHOSTS	1088
7.185.2.5 NUMBER_OF_SEND_GHOSTS	1088
7.186TYPES::DOMAIN_DOFS_TYPE Struct Reference	1089
7.186.1 Detailed Description	1089
7.186.2 Member Data Documentation	1089
7.186.2.1 DOF_INDEX	1089
7.186.2.2 DOMAIN	1089
7.186.2.3 NUMBER_OF_DOFS	1089
7.186.2.4 TOTAL_NUMBER_OF_DOFS	1090
7.187TYPES::DOMAIN_ELEMENT_TYPE Struct Reference	1091
7.187.1 Detailed Description	1091
7.187.2 Member Data Documentation	1091
7.187.2.1 BASIS	1091
7.187.2.2 ELEMENT_DERIVATIVES	1091
7.187.2.3 ELEMENT_NODES	1091
7.187.2.4 NUMBER	1092
7.188TYPES::DOMAIN_ELEMENTS_TYPE Struct Reference	1093

7.188.1 Detailed Description	1093
7.188.2 Member Data Documentation	1093
7.188.2.1 DOMAIN	1093
7.188.2.2 ELEMENTS	1093
7.188.2.3 MAXIMUM_NUMBER_OF_ELEMENT_PARAMETERS	1094
7.188.2.4 NUMBER_OF_ELEMENTS	1094
7.188.2.5 TOTAL_NUMBER_OF_ELEMENTS	1094
7.189 TYPES::DOMAIN_FACE_PTR_TYPE Struct Reference	1095
7.189.1 Detailed Description	1095
7.189.2 Member Data Documentation	1095
7.189.2.1 PTR	1095
7.190 TYPES::DOMAIN_FACE_TYPE Struct Reference	1096
7.190.1 Detailed Description	1096
7.190.2 Member Data Documentation	1096
7.190.2.1 BASIS	1096
7.190.2.2 DERIVATIVES_IN_FACE	1096
7.190.2.3 NODES_IN_FACE	1097
7.190.2.4 NUMBER	1097
7.190.2.5 XI_DIRECTION1	1097
7.190.2.6 XI_DIRECTION2	1097
7.191 TYPES::DOMAIN_FACES_TYPE Struct Reference	1098
7.191.1 Detailed Description	1098
7.191.2 Member Data Documentation	1098
7.191.2.1 DOMAIN	1098
7.191.2.2 FACES	1098
7.191.2.3 NUMBER_OF_FACES	1098
7.192 TYPES::DOMAIN_GLOBAL_MAPPING_TYPE Struct Reference	1099
7.192.1 Detailed Description	1099
7.192.2 Member Data Documentation	1099
7.192.2.1 DOMAIN_NUMBER	1099
7.192.2.2 LOCAL_NUMBER	1099
7.192.2.3 LOCAL_TYPE	1100
7.192.2.4 NUMBER_OF_DOMAINS	1100
7.193 TYPES::DOMAIN_LINE_PTR_TYPE Struct Reference	1101
7.193.1 Detailed Description	1101
7.193.2 Member Data Documentation	1101

7.193.2.1 PTR	1101
7.194 TYPES::DOMAIN_LINE_TYPE Struct Reference	1102
7.194.1 Detailed Description	1102
7.194.2 Member Data Documentation	1102
7.194.2.1 BASIS	1102
7.194.2.2 DERIVATIVES_IN_LINE	1102
7.194.2.3 NODES_IN_LINE	1102
7.194.2.4 NUMBER	1103
7.195 TYPES::DOMAIN_LINES_TYPE Struct Reference	1104
7.195.1 Detailed Description	1104
7.195.2 Member Data Documentation	1104
7.195.2.1 DOMAIN	1104
7.195.2.2 LINES	1104
7.195.2.3 NUMBER_OF_LINES	1104
7.196 TYPES::DOMAIN_MAPPING_TYPE Struct Reference	1105
7.196.1 Detailed Description	1106
7.196.2 Member Data Documentation	1106
7.196.2.1 ADJACENT_DOMAINS	1106
7.196.2.2 ADJACENT_DOMAINS_LIST	1106
7.196.2.3 ADJACENT_DOMAINS_PTR	1106
7.196.2.4 BOUNDARY_LIST	1107
7.196.2.5 GHOST_LIST	1107
7.196.2.6 GLOBAL_TO_LOCAL_MAP	1107
7.196.2.7 INTERNAL_LIST	1107
7.196.2.8 LOCAL_TO_GLOBAL_MAP	1107
7.196.2.9 NUMBER_OF_ADJACENT_DOMAINS	1107
7.196.2.10 NUMBER_OF_BOUNDARY	1108
7.196.2.11 NUMBER_OF_DOMAIN_LOCAL	1108
7.196.2.12 NUMBER_OF_DOMAINS	1108
7.196.2.13 NUMBER_OF_GHOST	1108
7.196.2.14 NUMBER_OF_GLOBAL	1108
7.196.2.15 NUMBER_OF_INTERNAL	1108
7.196.2.16 NUMBER_OF_LOCAL	1108
7.196.2.17 TOTAL_NUMBER_OF_LOCAL	1108
7.197 TYPES::DOMAIN_MAPPINGS_TYPE Struct Reference	1109
7.197.1 Detailed Description	1109

7.197.2 Member Data Documentation	1109
7.197.2.1 DOFS	1109
7.197.2.2 DOMAIN	1109
7.197.2.3 ELEMENTS	1109
7.197.2.4 NODES	1110
7.198TYPES::DOMAIN_NODE_TYPE Struct Reference	1111
7.198.1 Detailed Description	1111
7.198.2 Member Data Documentation	1112
7.198.2.1 DOF_INDEX	1112
7.198.2.2 GLOBAL_NUMBER	1112
7.198.2.3 LOCAL_NUMBER	1112
7.198.2.4 NODE_LINES	1112
7.198.2.5 NUMBER_OF_DERIVATIVES	1112
7.198.2.6 NUMBER_OF_NODE_LINES	1112
7.198.2.7 NUMBER_OF_SURROUNDING_ELEMENTS	1112
7.198.2.8 PARTIAL_DERIVATIVE_INDEX	1112
7.198.2.9 SURROUNDING_ELEMENTS	1113
7.198.2.10USER_NUMBER	1113
7.199TYPES::DOMAIN_NODES_TYPE Struct Reference	1114
7.199.1 Detailed Description	1114
7.199.2 Member Data Documentation	1114
7.199.2.1 DOMAIN	1114
7.199.2.2 MAXIMUM_NUMBER_OF_DERIVATIVES	1114
7.199.2.3 NODES	1114
7.199.2.4 NUMBER_OF_NODES	1115
7.199.2.5 TOTAL_NUMBER_OF_NODES	1115
7.200TYPES::DOMAIN_PTR_TYPE Struct Reference	1116
7.200.1 Detailed Description	1116
7.200.2 Member Data Documentation	1116
7.200.2.1 PTR	1116
7.201TYPES::DOMAIN_TOPOLOGY_TYPE Struct Reference	1117
7.201.1 Detailed Description	1117
7.201.2 Member Data Documentation	1117
7.201.2.1 DOFS	1117
7.201.2.2 DOMAIN	1117
7.201.2.3 ELEMENTS	1118

7.201.2.4 FACES	1118
7.201.2.5 LINES	1118
7.201.2.6 NODES	1118
7.202 TYPES::DOMAIN_TYPE Struct Reference	1119
7.202.1 Detailed Description	1119
7.202.2 Member Data Documentation	1119
7.202.2.1 DECOMPOSITION	1119
7.202.2.2 MAPPINGS	1120
7.202.2.3 MESH	1120
7.202.2.4 MESH_COMPONENT_NUMBER	1120
7.202.2.5 NODE_DOMAIN	1120
7.202.2.6 NUMBER_OF_DIMENSIONS	1120
7.202.2.7 REGION	1120
7.202.2.8 TOPOLOGY	1120
7.203 TYPES::EIGENPROBLEM_SOLVER_TYPE Struct Reference	1121
7.203.1 Detailed Description	1121
7.203.2 Member Data Documentation	1121
7.203.2.1 SOLVER	1121
7.203.2.2 SOLVER_LIBRARY	1121
7.204 TYPES::ELEMENT_MATRIX_TYPE Struct Reference	1122
7.204.1 Detailed Description	1122
7.204.2 Member Data Documentation	1122
7.204.2.1 COLUMN_DOFS	1122
7.204.2.2 EQUATIONS_MATRIX_NUMBER	1122
7.204.2.3 MATRIX	1122
7.204.2.4 MAX_NUMBER_OF_COLUMNS	1122
7.204.2.5 MAX_NUMBER_OF_ROWS	1122
7.204.2.6 NUMBER_OF_COLUMNS	1123
7.204.2.7 NUMBER_OF_ROWS	1123
7.204.2.8 ROW_DOFS	1123
7.205 TYPES::ELEMENT_VECTOR_TYPE Struct Reference	1124
7.205.1 Detailed Description	1124
7.205.2 Member Data Documentation	1124
7.205.2.1 MAX_NUMBER_OF_ROWS	1124
7.205.2.2 NUMBER_OF_ROWS	1124
7.205.2.3 ROW_DOFS	1124

7.205.2.4 VECTOR	1124
7.206TYPES::EQUATIONS_COL_TO_SOLVER_COLS_MAP_TYPE Struct Reference	1125
7.206.1 Detailed Description	1125
7.206.2 Member Data Documentation	1125
7.206.2.1 COUPLING_COEFFICIENTS	1125
7.206.2.2 NUMBER_OF_SOLVER_COLS	1125
7.206.2.3 SOLVER_COLS	1125
7.207TYPES::EQUATIONS_INTERPOLATION_TYPE Struct Reference	1126
7.207.1 Detailed Description	1127
7.207.2 Member Data Documentation	1127
7.207.2.1 DEPENDENT_FIELD	1127
7.207.2.2 DEPENDENT_INTERP_PARAMETERS	1127
7.207.2.3 DEPENDENT_INTERP_POINT	1127
7.207.2.4 EQUATIONS	1128
7.207.2.5 FIBRE_FIELD	1128
7.207.2.6 FIBRE_INTERP_PARAMETERS	1128
7.207.2.7 FIBRE_INTERP_POINT	1128
7.207.2.8 FIBRE_INTERP_POINT_METRICS	1128
7.207.2.9 GEOMETRIC_FIELD	1128
7.207.2.10GEOMETRIC_INTERP_PARAMETERS	1128
7.207.2.11GEOMETRIC_INTERP_POINT	1129
7.207.2.12GEOMETRIC_INTERP_POINT_METRICS	1129
7.207.2.13MATERIAL_FIELD	1129
7.207.2.14MATERIAL_INTERP_PARAMETERS	1129
7.207.2.15MATERIAL_INTERP_POINT	1129
7.207.2.16SOURCE_FIELD	1129
7.207.2.17SOURCE_INTERP_PARAMETERS	1129
7.207.2.18SOURCE_INTERP_POINT	1130
7.208TYPES::EQUATIONS_JACOBIAN_TO_VARIABLE_MAP_TYPE Struct Reference	1131
7.208.1 Detailed Description	1131
7.208.2 Member Data Documentation	1131
7.208.2.1 COLUMN_DOFS_MAPPING	1131
7.208.2.2 EQUATIONS_COLUMN_TO_DOF_VARIABLE_MAP	1131
7.208.2.3 JACOBIAN	1132
7.208.2.4 JACOBIAN_COEFFICIENT	1132
7.208.2.5 NUMBER_OF_COLUMNS	1132

7.208.2.6 VARIABLE	1132
7.208.2.7 VARIABLE_TYPE	1132
7.209TYPES::EQUATIONS_JACOBIAN_TYPE Struct Reference	1133
7.209.1 Detailed Description	1133
7.209.2 Member Data Documentation	1133
7.209.2.1 ELEMENT_JACOBIAN	1133
7.209.2.2 ELEMENT_RESIDUAL	1134
7.209.2.3 JACOBIAN	1134
7.209.2.4 JACOBIAN_CALCULATION_TYPE	1134
7.209.2.5 NONLINEAR_MATRICES	1134
7.209.2.6 NUMBER_OF_COLUMNS	1134
7.209.2.7 STORAGE_TYPE	1134
7.209.2.8 STRUCTURE_TYPE	1134
7.209.2.9 UPDATE_JACOBIAN	1135
7.210TYPES::EQUATIONS_LINEAR_DATA_TYPE Struct Reference	1136
7.210.1 Detailed Description	1136
7.210.2 Member Data Documentation	1136
7.210.2.1 EQUATIONS	1136
7.211TYPES::EQUATIONS_MAPPING_CREATE_VALUES_CACHE_TYPE Struct Reference	1137
7.211.1 Detailed Description	1137
7.211.2 Member Data Documentation	1137
7.211.2.1 MATRIX_COEFFICIENTS	1137
7.211.2.2 MATRIX_VARIABLE_TYPES	1138
7.211.2.3 NUMBER_OF_LINEAR_EQUATIONS_MATRICES	1138
7.211.2.4 RESIDUAL_COEFFICIENT	1138
7.211.2.5 RESIDUAL_VARIABLE_TYPE	1138
7.211.2.6 RHS_COEFFICIENT	1138
7.211.2.7 RHS_VARIABLE_TYPE	1138
7.211.2.8 SOURCE_COEFFICIENT	1138
7.211.2.9 SOURCE_VARIABLE_TYPE	1139
7.212TYPES::EQUATIONS_MAPPING_LINEAR_TYPE Struct Reference	1140
7.212.1 Detailed Description	1140
7.212.2 Member Data Documentation	1140
7.212.2.1 EQUATIONS_MAPPING	1140
7.212.2.2 EQUATIONS_MATRIX_TO_VARIABLE_MAPS	1141
7.212.2.3 EQUATIONS_ROW_TO_VARIABLE_DOF_MAPS	1141

7.212.2.4 MATRIX_VARIABLE_TYPES	1141
7.212.2.5 NUMBER_OF_LINEAR_EQUATIONS_MATRICES	1141
7.212.2.6 NUMBER_OF_LINEAR_MATRIX_VARIABLES	1141
7.212.2.7 VARIABLE_TO_EQUATIONS_MATRICES_MAPS	1141
7.213TYPES::EQUATIONS_MAPPING_NONLINEAR_TYPE Struct Reference	1142
7.213.1 Detailed Description	1142
7.213.2 Member Data Documentation	1142
7.213.2.1 EQUATIONS_MAPPING	1142
7.213.2.2 EQUATIONS_ROW_TO_RESIDUAL_DOF_MAP	1143
7.213.2.3 JACOBIAN_TO_VARIABLE_MAP	1143
7.213.2.4 RESIDUAL_COEFFICIENT	1143
7.213.2.5 RESIDUAL_DOF_TO_EQUATIONS_ROW_MAP	1143
7.213.2.6 RESIDUAL_VARIABLE	1143
7.213.2.7 RESIDUAL_VARIABLE_MAPPING	1143
7.213.2.8 RESIDUAL_VARIABLE_TYPE	1143
7.213.2.9 VARIABLE_TO_JACOBIAN_MAP	1144
7.214TYPES::EQUATIONS_MAPPING_RHS_TYPE Struct Reference	1145
7.214.1 Detailed Description	1145
7.214.2 Member Data Documentation	1145
7.214.2.1 EQUATIONS_MAPPING	1145
7.214.2.2 EQUATIONS_ROW_TO_RHS_DOF_MAP	1145
7.214.2.3 RHS_COEFFICIENT	1146
7.214.2.4 RHS_DOF_TO_EQUATIONS_ROW_MAP	1146
7.214.2.5 RHS_VARIABLE	1146
7.214.2.6 RHS_VARIABLE_MAPPING	1146
7.214.2.7 RHS_VARIABLE_TYPE	1146
7.215TYPES::EQUATIONS_MAPPING_SOURCE_TYPE Struct Reference	1147
7.215.1 Detailed Description	1147
7.215.2 Member Data Documentation	1147
7.215.2.1 EQUATIONS_MAPPING	1147
7.215.2.2 EQUATIONS_ROW_TO_SOURCE_DOF_MAP	1147
7.215.2.3 SOURCE_COEFFICIENT	1148
7.215.2.4 SOURCE_DOF_TO_EQUATIONS_ROW_MAP	1148
7.215.2.5 SOURCE_VARIABLE	1148
7.215.2.6 SOURCE_VARIABLE_MAPPING	1148
7.215.2.7 SOURCE_VARIABLE_TYPE	1148

7.216TYPES::EQUATIONS_MAPPING_TYPE Struct Reference	1149
7.216.1 Detailed Description	1149
7.216.2 Member Data Documentation	1150
7.216.2.1 CREATE_VALUES_CACHE	1150
7.216.2.2 EQUATIONS	1150
7.216.2.3 EQUATIONS_MAPPING_FINISHED	1150
7.216.2.4 EQUATIONS_MATRICES	1150
7.216.2.5 LINEAR_MAPPING	1150
7.216.2.6 NONLINEAR_MAPPING	1150
7.216.2.7 NUMBER_OF_GLOBAL_ROWS	1150
7.216.2.8 NUMBER_OF_ROWS	1150
7.216.2.9 RESIDUAL_VARIABLE	1151
7.216.2.10RESIDUAL_VARIABLE_MAPPING	1151
7.216.2.11RESIDUAL_VARIABLE_TYPE	1151
7.216.2.12RHS_MAPPING	1151
7.216.2.13ROW_DOFS_MAPPING	1151
7.216.2.14SOURCE_MAPPING	1151
7.216.2.15TOTAL_NUMBER_OF_ROWS	1151
7.217TYPES::EQUATIONS_MATRICES_LINEAR_TYPE Struct Reference	1152
7.217.1 Detailed Description	1152
7.217.2 Member Data Documentation	1152
7.217.2.1 EQUATIONS_MATRICES	1152
7.217.2.2 MATRICES	1152
7.217.2.3 NUMBER_OF_LINEAR_MATRICES	1152
7.218TYPES::EQUATIONS_MATRICES_NONLINEAR_TYPE Struct Reference	1153
7.218.1 Detailed Description	1153
7.218.2 Member Data Documentation	1153
7.218.2.1 ELEMENT_RESIDUAL	1153
7.218.2.2 EQUATIONS_MATRICES	1153
7.218.2.3 JACOBIAN	1153
7.218.2.4 RESIDUAL	1154
7.218.2.5 UPDATE_RESIDUAL	1154
7.219TYPES::EQUATIONS_MATRICES_RHS_TYPE Struct Reference	1155
7.219.1 Detailed Description	1155
7.219.2 Member Data Documentation	1155
7.219.2.1 ELEMENT_VECTOR	1155

7.219.2.2 EQUATIONS_MATRICES	1155
7.219.2.3 UPDATE_VECTOR	1155
7.219.2.4 VECTOR	1155
7.220TYPES::EQUATIONS_MATRICES_SOURCE_TYPE Struct Reference	1157
7.220.1 Detailed Description	1157
7.220.2 Member Data Documentation	1157
7.220.2.1 ELEMENT_VECTOR	1157
7.220.2.2 EQUATIONS_MATRICES	1157
7.220.2.3 UPDATE_VECTOR	1157
7.220.2.4 VECTOR	1157
7.221TYPES::EQUATIONS_MATRICES_TYPE Struct Reference	1159
7.221.1 Detailed Description	1159
7.221.2 Member Data Documentation	1159
7.221.2.1 EQUATIONS	1159
7.221.2.2 EQUATIONS_MAPPING	1160
7.221.2.3 EQUATIONS_MATRICES_FINISHED	1160
7.221.2.4 LINEAR_MATRICES	1160
7.221.2.5 NONLINEAR_MATRICES	1160
7.221.2.6 NUMBER_OF_GLOBAL_ROWS	1160
7.221.2.7 NUMBER_OF_ROWS	1160
7.221.2.8 RHS_VECTOR	1160
7.221.2.9 SOLUTION_MAPPING	1160
7.221.2.10 SOURCE_VECTOR	1161
7.221.2.11 TOTAL_NUMBER_OF_ROWS	1161
7.222TYPES::EQUATIONS_MATRIX_PTR_TYPE Struct Reference	1162
7.222.1 Detailed Description	1162
7.222.2 Member Data Documentation	1162
7.222.2.1 PTR	1162
7.223TYPES::EQUATIONS_MATRIX_TO_VARIABLE_MAP_TYPE Struct Reference	1163
7.223.1 Detailed Description	1163
7.223.2 Member Data Documentation	1163
7.223.2.1 COLUMN_DOFS_MAPPING	1163
7.223.2.2 COLUMN_TO_DOF_MAP	1164
7.223.2.3 EQUATIONS_MATRIX	1164
7.223.2.4 MATRIX_COEFFICIENT	1164
7.223.2.5 MATRIX_NUMBER	1164

7.223.2.6 NUMBER_OF_COLUMNS	1164
7.223.2.7 VARIABLE	1164
7.223.2.8 VARIABLE_TYPE	1164
7.224TYPES::EQUATIONS_MATRIX_TYPE Struct Reference	1165
7.224.1 Detailed Description	1165
7.224.2 Member Data Documentation	1165
7.224.2.1 ELEMENT_MATRIX	1165
7.224.2.2 LINEAR_MATRICES	1166
7.224.2.3 MATRIX	1166
7.224.2.4 MATRIX_NUMBER	1166
7.224.2.5 NUMBER_OF_COLUMNS	1166
7.224.2.6 STORAGE_TYPE	1166
7.224.2.7 STRUCTURE_TYPE	1166
7.224.2.8 UPDATE_MATRIX	1166
7.225TYPES::EQUATIONS_NONLINEAR_DATA_TYPE Struct Reference	1167
7.225.1 Detailed Description	1167
7.225.2 Member Data Documentation	1167
7.225.2.1 EQUATIONS	1167
7.226TYPES::EQUATIONS_ROW_TO_SOLVER_ROWS_MAP_TYPE Struct Reference . . .	1168
7.226.1 Detailed Description	1168
7.226.2 Member Data Documentation	1168
7.226.2.1 COUPLING_COEFFICIENTS	1168
7.226.2.2 NUMBER_OF_SOLVER_ROWS	1168
7.226.2.3 SOLVER_ROWS	1168
7.227TYPES::EQUATIONS_SET_ANALYTIC_TYPE Struct Reference	1169
7.227.1 Detailed Description	1169
7.227.2 Member Data Documentation	1169
7.227.2.1 ANALYTIC_FINISHED	1169
7.227.2.2 EQUATION_NUMBER	1169
7.227.2.3 EQUATIONS_SET	1169
7.228TYPES::EQUATIONS_SET_DEPENDENT_TYPE Struct Reference	1170
7.228.1 Detailed Description	1170
7.228.2 Member Data Documentation	1170
7.228.2.1 DEPENDENT_FIELD	1170
7.228.2.2 DEPENDENT_FINISHED	1170
7.228.2.3 EQUATIONS_SET	1170

7.229TYPES::EQUATIONS_SET_FIXED_CONDITIONS_TYPE Struct Reference	1171
7.229.1 Detailed Description	1171
7.229.2 Member Data Documentation	1171
7.229.2.1 BOUNDARY_CONDITIONS	1171
7.229.2.2 EQUATIONS_SET	1171
7.229.2.3 FIXED_CONDITIONS_FINISHED	1171
7.229.2.4 GLOBAL_BOUNDARY_CONDITIONS	1172
7.230TYPES::EQUATIONS_SET_GEOMETRY_TYPE Struct Reference	1173
7.230.1 Detailed Description	1173
7.230.2 Member Data Documentation	1173
7.230.2.1 EQUATIONS_SET	1173
7.230.2.2 FIBRE_FIELD	1173
7.230.2.3 GEOMETRIC_FIELD	1173
7.231TYPES::EQUATIONS_SET_MATERIALS_TYPE Struct Reference	1174
7.231.1 Detailed Description	1174
7.231.2 Member Data Documentation	1174
7.231.2.1 EQUATIONS_SET	1174
7.231.2.2 MATERIAL_FIELD	1174
7.231.2.3 MATERIALS_FINISHED	1174
7.232TYPES::EQUATIONS_SET_PTR_TYPE Struct Reference	1175
7.232.1 Detailed Description	1175
7.232.2 Member Data Documentation	1175
7.232.2.1 PTR	1175
7.233TYPES::EQUATIONS_SET_SOURCE_TYPE Struct Reference	1176
7.233.1 Detailed Description	1176
7.233.2 Member Data Documentation	1176
7.233.2.1 EQUATIONS_SET	1176
7.233.2.2 SOURCE_FIELD	1176
7.233.2.3 SOURCE_FINISHED	1176
7.234TYPES::EQUATIONS_SET_TO_SOLVER_MAP_TYPE Struct Reference	1177
7.234.1 Detailed Description	1177
7.234.2 Member Data Documentation	1177
7.234.2.1 EQUATIONS	1177
7.234.2.2 EQUATIONS_ROW_TO_SOLVER_ROWS_MAPS	1178
7.234.2.3 EQUATIONS_SET_INDEX	1178
7.234.2.4 EQUATIONS_TO_SOLVER_MATRIX_MAPS_EM	1178

7.234.2.5 EQUATIONS_TO_SOLVER_MATRIX_MAPS_JM	1178
7.234.2.6 EQUATIONS_TO_SOLVER_MATRIX_MAPS_SM	1178
7.234.2.7 SOLUTION_MAPPING	1178
7.235TYPES::EQUATIONS_SET_TYPE Struct Reference	1179
7.235.1 Detailed Description	1180
7.235.2 Member Data Documentation	1180
7.235.2.1 ANALYTIC	1180
7.235.2.2 CLASS	1180
7.235.2.3 DEPENDENT	1180
7.235.2.4 EQUATIONS	1180
7.235.2.5 EQUATIONS_SET_FINISHED	1180
7.235.2.6 EQUATIONS_SETS	1181
7.235.2.7 FIXED_CONDITIONS	1181
7.235.2.8 GEOMETRY	1181
7.235.2.9 GLOBAL_NUMBER	1181
7.235.2.10LINEARITY	1181
7.235.2.11MATERIALS	1181
7.235.2.12REGION	1181
7.235.2.13SOLUTION_METHOD	1182
7.235.2.14SOURCE	1182
7.235.2.15SUBTYPE	1182
7.235.2.16TIME_TYPE	1182
7.235.2.17TYPE	1182
7.235.2.18USER_NUMBER	1182
7.236TYPES::EQUATIONS_SETS_TYPE Struct Reference	1183
7.236.1 Detailed Description	1183
7.236.2 Member Data Documentation	1183
7.236.2.1 EQUATIONS_SETS	1183
7.236.2.2 NUMBER_OF_EQUATIONS_SETS	1183
7.236.2.3 REGION	1183
7.237TYPES::EQUATIONS_TIME_DATA_TYPE Struct Reference	1184
7.237.1 Detailed Description	1184
7.237.2 Member Data Documentation	1184
7.237.2.1 EQUATIONS	1184
7.238TYPES::EQUATIONS_TO_SOLVER_MAPS_PTR_TYPE Struct Reference	1185
7.238.1 Detailed Description	1185

7.238.2 Member Data Documentation	1185
7.238.2.1 PTR	1185
7.239TYPES::EQUATIONS_TO_SOLVER_MAPS_TYPE Struct Reference	1186
7.239.1 Detailed Description	1186
7.239.2 Member Data Documentation	1186
7.239.2.1 EQUATIONS_COL_SOLVER_COLS_MAP	1186
7.239.2.2 EQUATIONS_MATRIX	1186
7.239.2.3 EQUATIONS_MATRIX_NUMBER	1187
7.239.2.4 SOLVER_MATRIX	1187
7.239.2.5 SOLVER_MATRIX_NUMBER	1187
7.240TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_EM_TYPE Struct Reference . .	1188
7.240.1 Detailed Description	1188
7.240.2 Member Data Documentation	1188
7.240.2.1 EQUATIONS_MATRIX_NUMBER	1188
7.240.2.2 EQUATIONS_TO_SOLVER_MATRIX_MAPS	1188
7.240.2.3 NUMBER_OF_SOLVER_MATRICES	1188
7.241TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_JM_TYPE Struct Reference . .	1190
7.241.1 Detailed Description	1190
7.241.2 Member Data Documentation	1190
7.241.2.1 JACOBIAN_TO_SOLVER_MATRIX_MAP	1190
7.242TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_SM_TYPE Struct Reference . .	1191
7.242.1 Detailed Description	1191
7.242.2 Member Data Documentation	1192
7.242.2.1 EQUATIONS_TO_SOLVER_MATRIX_MAPS	1192
7.242.2.2 JACOBIAN_TO_SOLVER_MATRIX_MAP	1192
7.242.2.3 NUMBER_OF_LINEAR_EQUATIONS_MATRICES	1192
7.242.2.4 NUMBER_OF_VARIABLES	1192
7.242.2.5 SOLVER_MATRIX_NUMBER	1192
7.242.2.6 VARIABLE_TO_SOLVER_COL_MAPS	1192
7.242.2.7 VARIABLE_TYPES	1192
7.242.2.8 VARIABLES	1193
7.243TYPES::EQUATIONS_TYPE Struct Reference	1194
7.243.1 Detailed Description	1194
7.243.2 Member Data Documentation	1195
7.243.2.1 EQUATIONS_FINISHED	1195
7.243.2.2 EQUATIONS_MAPPING	1195

7.243.2.3 EQUATIONS_MATRICES	1195
7.243.2.4 EQUATIONS_SET	1195
7.243.2.5 INTERPOLATION	1195
7.243.2.6 LINEAR_DATA	1195
7.243.2.7 NONLINEAR_DATA	1195
7.243.2.8 NONLINEAR_JACOBIAN_TYPE	1196
7.243.2.9 OUTPUT_TYPE	1196
7.243.2.10 SPARSITY_TYPE	1196
7.243.2.11 TIME_DATA	1196
7.244 TYPES::FIELD_CREATE_VALUES_CACHE_TYPE Struct Reference	1197
7.244.1 Detailed Description	1197
7.244.2 Member Data Documentation	1197
7.244.2.1 INTERPOLATION_TYPE	1197
7.244.2.2 MESH_COMPONENT_NUMBER	1197
7.244.2.3 NUMBER_OF_COMPONENTS	1198
7.244.2.4 VARIABLE_TYPES	1198
7.245 TYPES::FIELD_DOF_TO_PARAM_MAP_TYPE Struct Reference	1199
7.245.1 Detailed Description	1200
7.245.2 Member Data Documentation	1200
7.245.2.1 CONSTANT_DOF2PARAM_MAP	1200
7.245.2.2 DOF_TYPE	1200
7.245.2.3 ELEMENT_DOF2PARAM_MAP	1200
7.245.2.4 NODE_DOF2PARAM_MAP	1201
7.245.2.5 NUMBER_OF_CONSTANT_DOFS	1201
7.245.2.6 NUMBER_OF_DOFS	1201
7.245.2.7 NUMBER_OF_ELEMENT_DOFS	1201
7.245.2.8 NUMBER_OF_NODE_DOFS	1201
7.245.2.9 NUMBER_OF_POINT_DOFS	1201
7.245.2.10 POINT_DOF2PARAM_MAP	1201
7.245.2.11 VARIABLE_DOF	1202
7.246 TYPES::FIELD_GEOMETRIC_PARAMETERS_TYPE Struct Reference	1203
7.246.1 Detailed Description	1203
7.246.2 Member Data Documentation	1203
7.246.2.1 AREAS	1203
7.246.2.2 FIELDS_USING	1204
7.246.2.3 LENGTHS	1204

7.246.2.4 NUMBER_OF AREAS	1204
7.246.2.5 NUMBER_OF_FIELDS_USING	1204
7.246.2.6 NUMBER_OF_LINES	1204
7.246.2.7 NUMBER_OF_VOLUMES	1204
7.246.2.8 VOLUMES	1204
7.247TYPES::FIELD_INTERPOLATED_POINT_METRICS_TYPE Struct Reference	1205
7.247.1 Detailed Description	1205
7.247.2 Member Data Documentation	1205
7.247.2.1 DX_DXI	1205
7.247.2.2 DXI_DX	1206
7.247.2.3 GL	1206
7.247.2.4 GU	1206
7.247.2.5 INTERPOLATED_POINT	1206
7.247.2.6 JACOBIAN	1206
7.247.2.7 JACOBIAN_TYPE	1206
7.247.2.8 NUMBER_OF_X_DIMENSIONS	1206
7.247.2.9 NUMBER_OF_XI_DIMENSIONS	1207
7.248TYPES::FIELD_INTERPOLATED_POINT_TYPE Struct Reference	1208
7.248.1 Detailed Description	1208
7.248.2 Member Data Documentation	1208
7.248.2.1 INTERPOLATION_PARAMETERS	1208
7.248.2.2 MAX_PARTIAL_DERIVATIVE_INDEX	1208
7.248.2.3 PARTIAL_DERIVATIVE_TYPE	1209
7.248.2.4 VALUES	1209
7.249TYPES::FIELD_INTERPOLATION_PARAMETERS_TYPE Struct Reference	1210
7.249.1 Detailed Description	1210
7.249.2 Member Data Documentation	1210
7.249.2.1 BASES	1210
7.249.2.2 FIELD	1210
7.249.2.3 FIELD_VARIABLE	1211
7.249.2.4 NUMBER_OF_PARAMETERS	1211
7.249.2.5 PARAMETERS	1211
7.250TYPES::FIELD_MAPPINGS_TYPE Struct Reference	1212
7.250.1 Detailed Description	1212
7.250.2 Member Data Documentation	1212
7.250.2.1 DOF_TO_PARAM_MAP	1212

7.250.2.2 DOMAIN_MAPPING	1212
7.251TYPES::FIELD_PARAM_TO_DOF_MAP_TYPE Struct Reference	1213
7.251.1 Detailed Description	1213
7.251.2 Member Data Documentation	1214
7.251.2.1 CONSTANT_PARAM2DOF_MAP	1214
7.251.2.2 ELEMENT_PARAM2DOF_MAP	1214
7.251.2.3 MAX_NUMBER_OF_DERIVATIVES	1214
7.251.2.4 NODE_PARAM2DOF_MAP	1214
7.251.2.5 NUMBER_OF_CONSTANT_PARAMETERS	1214
7.251.2.6 NUMBER_OF_ELEMENT_PARAMETERS	1215
7.251.2.7 NUMBER_OF_NODE_PARAMETERS	1215
7.251.2.8 NUMBER_OF_POINT_PARAMETERS	1215
7.251.2.9 POINT_PARAM2DOF_MAP	1215
7.252TYPES::FIELD_PARAMETER_SET_PTR_TYPE Struct Reference	1216
7.252.1 Detailed Description	1216
7.252.2 Member Data Documentation	1216
7.252.2.1 PTR	1216
7.253TYPES::FIELD_PARAMETER_SET_TYPE Struct Reference	1217
7.253.1 Detailed Description	1217
7.253.2 Member Data Documentation	1217
7.253.2.1 PARAMETERS	1217
7.253.2.2 SET_INDEX	1217
7.253.2.3 SET_TYPE	1217
7.254TYPES::FIELD_PARAMETER_SETS_TYPE Struct Reference	1219
7.254.1 Detailed Description	1219
7.254.2 Member Data Documentation	1219
7.254.2.1 FIELD	1219
7.254.2.2 NUMBER_OF_PARAMETER_SETS	1219
7.254.2.3 PARAMETER_SETS	1220
7.254.2.4 SET_TYPE	1220
7.255TYPES::FIELD_PTR_TYPE Struct Reference	1221
7.255.1 Detailed Description	1221
7.255.2 Member Data Documentation	1221
7.255.2.1 PTR	1221
7.256TYPES::FIELD_SCALING_TYPE Struct Reference	1222
7.256.1 Detailed Description	1222

7.256.2 Member Data Documentation	1222
7.256.2.1 MAX_NUMBER_OF_DERIVATIVES	1222
7.256.2.2 MAX_NUMBER_OF_ELEMENT_PARAMETERS	1222
7.256.2.3 MESH_COMPONENT_NUMBER	1222
7.256.2.4 SCALE_FACTORS	1223
7.257TYPES::FIELD_SCALINGS_TYPE Struct Reference	1224
7.257.1 Detailed Description	1224
7.257.2 Member Data Documentation	1224
7.257.2.1 NUMBER_OF_SCALING_INDICES	1224
7.257.2.2 SCALING_TYPE	1224
7.257.2.3 SCALINGS	1224
7.258TYPES::FIELD_TYPE Struct Reference	1225
7.258.1 Detailed Description	1226
7.258.2 Member Data Documentation	1226
7.258.2.1 DECOMPOSITION	1226
7.258.2.2 DEPENDENT_TYPE	1226
7.258.2.3 DIMENSION	1226
7.258.2.4 FIELD_FINISHED	1227
7.258.2.5 FIELDS	1227
7.258.2.6 GEOMETRIC_FIELD	1227
7.258.2.7 GEOMETRIC_FIELD_PARAMETERS	1227
7.258.2.8 GLOBAL_NUMBER	1227
7.258.2.9 MAPPINGS	1227
7.258.2.10NUMBER_OF_VARIABLES	1227
7.258.2.11PARAMETER_SETS	1227
7.258.2.12REGION	1228
7.258.2.13SCALINGS	1228
7.258.2.14TYPE	1228
7.258.2.15USER_NUMBER	1228
7.258.2.16VARIABLE_TYPE_MAP	1228
7.258.2.17VARIABLES	1228
7.259TYPES::FIELD_VARIABLE_COMPONENT_TYPE Struct Reference	1229
7.259.1 Detailed Description	1229
7.259.2 Member Data Documentation	1230
7.259.2.1 COMPONENT_NUMBER	1230
7.259.2.2 DOMAIN	1230

7.259.2.3 FIELD	1230
7.259.2.4 FIELD_VARIABLE	1230
7.259.2.5 INTERPOLATION_TYPE	1230
7.259.2.6 MAX_NUMBER_OF_INTERPOLATION_PARAMETERS	1230
7.259.2.7 MESH_COMPONENT_NUMBER	1230
7.259.2.8 PARAM_TO_DOF_MAP	1231
7.259.2.9 REGION	1231
7.259.2.10 SCALING_INDEX	1231
7.260 TYPES::FIELD_VARIABLE_PTR_TYPE Struct Reference	1232
7.260.1 Detailed Description	1232
7.260.2 Member Data Documentation	1232
7.260.2.1 PTR	1232
7.261 TYPES::FIELD_VARIABLE_TYPE Struct Reference	1233
7.261.1 Detailed Description	1234
7.261.2 Member Data Documentation	1234
7.261.2.1 COMPONENTS	1234
7.261.2.2 DOF_LIST	1234
7.261.2.3 DOMAIN_MAPPING	1234
7.261.2.4 FIELD	1234
7.261.2.5 GLOBAL_DOF_OFFSET	1234
7.261.2.6 MAX_NUMBER_OF_INTERPOLATION_PARAMETERS	1234
7.261.2.7 NUMBER_OF_COMPONENTS	1235
7.261.2.8 NUMBER_OF_DOFS	1235
7.261.2.9 NUMBER_OF_GLOBAL_DOFS	1235
7.261.2.10 REGION	1235
7.261.2.11 TOTAL_NUMBER_OF_DOFS	1235
7.261.2.12 VARIABLE_NUMBER	1235
7.261.2.13 VARIABLE_TYPE	1235
7.262 TYPES::FIELDS_TYPE Struct Reference	1236
7.262.1 Detailed Description	1236
7.262.2 Member Data Documentation	1236
7.262.2.1 FIELDS	1236
7.262.2.2 NUMBER_OF_FIELDS	1236
7.262.2.3 REGION	1236
7.263 TYPES::GENERATED_MESH_PTR_TYPE Struct Reference	1237
7.263.1 Detailed Description	1237

7.263.2 Member Data Documentation	1237
7.263.2.1 PTR	1237
7.264TYPES::GENERATED_MESH_REGULAR_TYPE Struct Reference	1238
7.264.1 Detailed Description	1238
7.264.2 Member Data Documentation	1238
7.264.2.1 BASIS	1238
7.264.2.2 GENERATED_MESH	1238
7.264.2.3 MAXIMUM_EXTENT	1239
7.264.2.4 MESH_DIMENSION	1239
7.264.2.5 NUMBER_OF_ELEMENTS_XI	1239
7.264.2.6 ORIGIN	1239
7.265TYPES::GENERATED_MESH_TYPE Struct Reference	1240
7.265.1 Detailed Description	1240
7.265.2 Member Data Documentation	1240
7.265.2.1 GENERATED_MESH_FINISHED	1240
7.265.2.2 GENERATED_TYPE	1240
7.265.2.3 GLOBAL_NUMBER	1240
7.265.2.4 MESH	1240
7.265.2.5 REGION	1240
7.265.2.6 REGULAR_MESH	1241
7.265.2.7 USER_NUMBER	1241
7.266TYPES::GENERATED_MESSES_TYPE Struct Reference	1242
7.266.1 Detailed Description	1242
7.266.2 Member Data Documentation	1242
7.266.2.1 GENERATED_MESSES	1242
7.266.2.2 NUMBER_OF_GENERATED_MESSES	1242
7.267TYPES::JACOBIAN_COL_TO_SOLVER_COLS_MAP_TYPE Struct Reference	1243
7.267.1 Detailed Description	1243
7.267.2 Member Data Documentation	1243
7.267.2.1 COUPLING_COEFFICIENTS	1243
7.267.2.2 NUMBER_OF_SOLVER_COLS	1243
7.267.2.3 SOLVER_COLS	1243
7.268TYPES::JACOBIAN_TO_SOLVER_MAP_TYPE Struct Reference	1244
7.268.1 Detailed Description	1244
7.268.2 Member Data Documentation	1244
7.268.2.1 JACOBIAN_COL_SOLVER_COLS_MAP	1244

7.268.2.2 JACOBIAN_MATRIX	1244
7.268.2.3 SOLVER_MATRIX	1244
7.268.2.4 SOLVER_MATRIX_NUMBER	1245
7.269TYPES::LINEAR_DIRECT_SOLVER_TYPE Struct Reference	1246
7.269.1 Detailed Description	1246
7.269.2 Member Data Documentation	1246
7.269.2.1 DIRECT_SOLVER_TYPE	1246
7.269.2.2 LINEAR_SOLVER	1246
7.269.2.3 SOLVER_LIBRARY	1246
7.270TYPES::LINEAR_ITERATIVE_SOLVER_TYPE Struct Reference	1247
7.270.1 Detailed Description	1247
7.270.2 Member Data Documentation	1248
7.270.2.1 ABSOLUTE_TOLERANCE	1248
7.270.2.2 DIVERGENCE_TOLERANCE	1248
7.270.2.3 ITERATIVE_PRECONDITIONER_TYPE	1248
7.270.2.4 ITERATIVE_SOLVER_TYPE	1248
7.270.2.5 KSP	1248
7.270.2.6 LINEAR_SOLVER	1248
7.270.2.7 MAXIMUM_NUMBER_OF_ITERATIONS	1248
7.270.2.8 PC	1249
7.270.2.9 RELATIVE_TOLERANCE	1249
7.270.2.10 SOLVER_LIBRARY	1249
7.271TYPES::LINEAR_SOLVER_TYPE Struct Reference	1250
7.271.1 Detailed Description	1250
7.271.2 Member Data Documentation	1250
7.271.2.1 DIRECT_SOLVER	1250
7.271.2.2 ITERATIVE_SOLVER	1250
7.271.2.3 LINEAR_SOLVE_TYPE	1250
7.271.2.4 SOLVER	1251
7.272TYPES::MATRIX_TYPE Struct Reference	1252
7.272.1 Detailed Description	1253
7.272.2 Member Data Documentation	1253
7.272.2.1 COLUMN_INDICES	1253
7.272.2.2 DATA_DP	1253
7.272.2.3 DATA_INTG	1253
7.272.2.4 DATA_L	1253

7.272.2.5 DATA_SP	1254
7.272.2.6 DATA_TYPE	1254
7.272.2.7 ID	1254
7.272.2.8 M	1254
7.272.2.9 MATRIX_FINISHED	1254
7.272.2.10 MAX_M	1254
7.272.2.11 IMAX_N	1254
7.272.2.12 MAXIMUM_COLUMN_INDICES_PER_ROW	1254
7.272.2.13 N	1255
7.272.2.14 NUMBER_NON_ZEROS	1255
7.272.2.15 ROW_INDICES	1255
7.272.2.16 SIZE	1255
7.272.2.17 STORAGE_TYPE	1255
7.273 TYPES::MESH_DOFS_TYPE Struct Reference	1256
7.273.1 Detailed Description	1256
7.273.2 Member Data Documentation	1256
7.273.2.1 MESH	1256
7.273.2.2 NUMBER_OF_DOFS	1256
7.274 TYPES::MESH_ELEMENT_TYPE Struct Reference	1257
7.274.1 Detailed Description	1257
7.274.2 Member Data Documentation	1257
7.274.2.1 ADJACENT_ELEMENTS	1257
7.274.2.2 BASIS	1258
7.274.2.3 GLOBAL_ELEMENT_NODES	1258
7.274.2.4 GLOBAL_NUMBER	1258
7.274.2.5 NUMBER_OF_ADJACENT_ELEMENTS	1258
7.274.2.6 USER_ELEMENT_NODES	1258
7.274.2.7 USER_NUMBER	1258
7.275 TYPES::MESH_ELEMENTS_TYPE Struct Reference	1259
7.275.1 Detailed Description	1259
7.275.2 Member Data Documentation	1259
7.275.2.1 ELEMENTS	1259
7.275.2.2 ELEMENTS_FINISHED	1259
7.275.2.3 MESH	1260
7.275.2.4 NUMBER_OF_ELEMENTS	1260
7.276 TYPES::MESH_NODE_TYPE Struct Reference	1261

7.276.1 Detailed Description	1261
7.276.2 Member Data Documentation	1261
7.276.2.1 DOF_INDEX	1261
7.276.2.2 GLOBAL_NUMBER	1261
7.276.2.3 NUMBER_OF_DERIVATIVES	1262
7.276.2.4 NUMBER_OF_SURROUNDING_ELEMENTS	1262
7.276.2.5 PARTIAL_DERIVATIVE_INDEX	1262
7.276.2.6 SURROUNDING_ELEMENTS	1262
7.276.2.7 USER_NUMBER	1262
7.277TYPES::MESH_NODES_TYPE Struct Reference	1263
7.277.1 Detailed Description	1263
7.277.2 Member Data Documentation	1263
7.277.2.1 MESH	1263
7.277.2.2 NODES	1263
7.277.2.3 NUMBER_OF_NODES	1263
7.278TYPES::MESH_PTR_TYPE Struct Reference	1264
7.278.1 Detailed Description	1264
7.278.2 Member Data Documentation	1264
7.278.2.1 PTR	1264
7.279TYPES::MESH_TOPOLOGY_PTR_TYPE Struct Reference	1265
7.279.1 Detailed Description	1265
7.279.2 Member Data Documentation	1265
7.279.2.1 PTR	1265
7.280TYPES::MESH_TOPOLOGY_TYPE Struct Reference	1266
7.280.1 Detailed Description	1266
7.280.2 Member Data Documentation	1266
7.280.2.1 DOFS	1266
7.280.2.2 ELEMENTS	1266
7.280.2.3 MESH	1266
7.280.2.4 MESH_COMPONENT_NUMBER	1267
7.280.2.5 NODES	1267
7.281TYPES::MESH_TYPE Struct Reference	1268
7.281.1 Detailed Description	1269
7.281.2 Member Data Documentation	1269
7.281.2.1 DECOMPOSITIONS	1269
7.281.2.2 EMBEDDED_MESHES	1269

7.281.2.3 EMBEDDING_MESH	1269
7.281.2.4 GENERATED_MESH	1269
7.281.2.5 GLOBAL_NUMBER	1269
7.281.2.6 MESH_EMBEDDED	1270
7.281.2.7 MESH_FINISHED	1270
7.281.2.8 MESHES	1270
7.281.2.9 NUMBER_OF_COMPONENTS	1270
7.281.2.10 NUMBER_OF_DIMENSIONS	1270
7.281.2.11 NUMBER_OF_ELEMENTS	1270
7.281.2.12 NUMBER_OF_EMBEDDED_MESHES	1270
7.281.2.13 NUMBER_OF_FACES	1270
7.281.2.14 NUMBER_OF_LINES	1270
7.281.2.15 REGION	1271
7.281.2.16 TOPOLOGY	1271
7.281.2.17 USER_NUMBER	1271
7.282 TYPES::MESHES_TYPE Struct Reference	1272
7.282.1 Detailed Description	1272
7.282.2 Member Data Documentation	1272
7.282.2.1 MESHES	1272
7.282.2.2 NUMBER_OF_MESHES	1272
7.282.2.3 REGION	1272
7.283 TYPES::NODE_TYPE Struct Reference	1273
7.283.1 Detailed Description	1273
7.283.2 Member Data Documentation	1273
7.283.2.1 GLOBAL_NUMBER	1273
7.283.2.2 INITIAL_POSITION	1273
7.283.2.3 LABEL	1273
7.283.2.4 USER_NUMBER	1274
7.284 TYPES::NODES_TYPE Struct Reference	1275
7.284.1 Detailed Description	1275
7.284.2 Member Data Documentation	1275
7.284.2.1 NODE_TREE	1275
7.284.2.2 NODES	1275
7.284.2.3 NODES_FINISHED	1276
7.284.2.4 NUMBER_OF_NODES	1276
7.284.2.5 REGION	1276

7.285TYPES::NONLINEAR_LINESEARCH_SOLVER_TYPE Struct Reference	1277
7.285.1 Detailed Description	1277
7.285.2 Member Data Documentation	1278
7.285.2.1 JACOBIAN_CALCULATION_TYPE	1278
7.285.2.2 JACOBIAN_FDCOLORING	1278
7.285.2.3 JACOBIAN_ISCOLORING	1278
7.285.2.4 LINESEARCH_ALPHA	1278
7.285.2.5 LINESEARCH_MAXSTEP	1278
7.285.2.6 LINESEARCH_STEPTOLERANCE	1278
7.285.2.7 LINESEARCH_TYPE	1278
7.285.2.8 NONLINEAR_SOLVER	1279
7.285.2.9 SNES	1279
7.285.2.10 SOLVER_LIBRARY	1279
7.286TYPES::NONLINEAR_SOLVER_TYPE Struct Reference	1280
7.286.1 Detailed Description	1280
7.286.2 Member Data Documentation	1280
7.286.2.1 ABSOLUTE_TOLERANCE	1280
7.286.2.2 LINESEARCH_SOLVER	1281
7.286.2.3 MAXIMUM_NUMBER_OF_FUNCTION_EVALUATIONS	1281
7.286.2.4 MAXIMUM_NUMBER_OF_ITERATIONS	1281
7.286.2.5 NONLINEAR_SOLVE_TYPE	1281
7.286.2.6 RELATIVE_TOLERANCE	1281
7.286.2.7 SOLUTION_TOLERANCE	1281
7.286.2.8 SOLVER	1281
7.286.2.9 TRUSTREGION_SOLVER	1282
7.287TYPES::NONLINEAR_TRUSTREGION_SOLVER_TYPE Struct Reference	1283
7.287.1 Detailed Description	1283
7.287.2 Member Data Documentation	1283
7.287.2.1 NONLINEAR_SOLVER	1283
7.287.2.2 SNES	1283
7.287.2.3 SOLVER_LIBRARY	1284
7.287.2.4 TRUSTREGION_DELTA0	1284
7.287.2.5 TRUSTREGION_TOLERANCE	1284
7.288TYPES::PROBLEM_CONTROL_TYPE Struct Reference	1285
7.288.1 Detailed Description	1285
7.288.2 Member Data Documentation	1285

7.288.2.1 CONTROL_FINISHED	1285
7.288.2.2 CONTROL_TYPE	1285
7.288.2.3 PROBLEM	1285
7.289 TYPES::PROBLEM_LINEAR_DATA_TYPE Struct Reference	1286
7.289.1 Detailed Description	1286
7.289.2 Member Data Documentation	1286
7.289.2.1 SOLUTION	1286
7.290 TYPES::PROBLEM_NONLINEAR_DATA_TYPE Struct Reference	1287
7.290.1 Detailed Description	1287
7.290.2 Member Data Documentation	1287
7.290.2.1 NUMBER_OF_ITERATIONS	1287
7.290.2.2 SOLUTION	1287
7.291 TYPES::PROBLEM_PTR_TYPE Struct Reference	1288
7.291.1 Detailed Description	1288
7.291.2 Member Data Documentation	1288
7.291.2.1 PTR	1288
7.292 TYPES::PROBLEM_TIME_DATA_TYPE Struct Reference	1289
7.292.1 Detailed Description	1289
7.292.2 Member Data Documentation	1289
7.292.2.1 SOLUTION	1289
7.293 TYPES::PROBLEM_TYPE Struct Reference	1290
7.293.1 Detailed Description	1290
7.293.2 Member Data Documentation	1290
7.293.2.1 CLASS	1290
7.293.2.2 CONTROL	1291
7.293.2.3 GLOBAL_NUMBER	1291
7.293.2.4 NUMBER_OF_SOLUTIONS	1291
7.293.2.5 PROBLEM_FINISHED	1291
7.293.2.6 PROBLEMS	1291
7.293.2.7 SOLUTIONS	1291
7.293.2.8 SUBTYPE	1291
7.293.2.9 TYPE	1291
7.293.2.10 USER_NUMBER	1292
7.294 TYPES::PROBLEMS_TYPE Struct Reference	1293
7.294.1 Detailed Description	1293
7.294.2 Member Data Documentation	1293

7.294.2.1 NUMBER_OF_PROBLEMS	1293
7.294.2.2 PROBLEMS	1293
7.295TYPES::QUADRATURE_SCHEME_PTR_TYPE Struct Reference	1294
7.295.1 Detailed Description	1294
7.295.2 Member Data Documentation	1294
7.295.2.1 PTR	1294
7.296TYPES::QUADRATURE_SCHEME_TYPE Struct Reference	1295
7.296.1 Detailed Description	1295
7.296.2 Member Data Documentation	1295
7.296.2.1 GAUSS_BASIS_FNS	1295
7.296.2.2 GAUSS_POSITIONS	1296
7.296.2.3 GAUSS_WEIGHTS	1296
7.296.2.4 GLOBAL_NUMBER	1296
7.296.2.5 NUMBER_OF_GAUSS	1296
7.296.2.6 QUADRATURE	1296
7.297TYPES::QUADRATURE_TYPE Struct Reference	1297
7.297.1 Detailed Description	1297
7.297.2 Member Data Documentation	1297
7.297.2.1 BASIS	1297
7.297.2.2 GAUSS_ORDER	1298
7.297.2.3 NUMBER_OF_GAUSS_XI	1298
7.297.2.4 NUMBER_OF_SCHEMES	1298
7.297.2.5 QUADRATURE_SCHEME_MAP	1298
7.297.2.6 SCHEMES	1298
7.297.2.7 TYPE	1298
7.298TYPES::REGION_PTR_TYPE Struct Reference	1299
7.298.1 Detailed Description	1299
7.298.2 Member Data Documentation	1299
7.298.2.1 PTR	1299
7.299TYPES::REGION_TYPE Struct Reference	1300
7.299.1 Detailed Description	1300
7.299.2 Member Data Documentation	1301
7.299.2.1 COORDINATE_SYSTEM	1301
7.299.2.2 EQUATIONS_SETS	1301
7.299.2.3 FIELDS	1301
7.299.2.4 LABEL	1301

7.299.2.5 MESHES	1301
7.299.2.6 NODES	1301
7.299.2.7 NUMBER_OF_SUB_REGIONS	1301
7.299.2.8 PARENT_REGION	1301
7.299.2.9 REGION_FINISHED	1302
7.299.2.10SUB_REGIONS	1302
7.299.2.11USER_NUMBER	1302
7.300TYPES::SOLUTION_MAPPING_CREATE_VALUES_CACHE_TYPE Struct Reference	1303
7.300.1 Detailed Description	1303
7.300.2 Member Data Documentation	1303
7.300.2.1 MATRIX_VARIABLE_TYPES	1303
7.300.2.2 RESIDUAL_VARIABLE_TYPE	1303
7.300.2.3 RHS_VARIABLE_TYPE	1304
7.300.2.4 SOURCE_VARIABLE_TYPE	1304
7.301TYPES::SOLUTION_MAPPING_TYPE Struct Reference	1305
7.301.1 Detailed Description	1306
7.301.2 Member Data Documentation	1306
7.301.2.1 CREATE_VALUES_CACHE	1306
7.301.2.2 EQUATIONS_SET_TO_SOLVER_MAP	1306
7.301.2.3 EQUATIONS_SETS	1306
7.301.2.4 HAVE_JACOBIAN	1306
7.301.2.5 NUMBER_OF_EQUATIONS_SETS	1306
7.301.2.6 NUMBER_OF_GLOBAL_ROWS	1306
7.301.2.7 NUMBER_OF_ROWS	1307
7.301.2.8 NUMBER_OF_SOLVER_MATRICES	1307
7.301.2.9 ROW_DOFS_MAPPING	1307
7.301.2.10SOLUTION	1307
7.301.2.11ISOLUTION_MAPPING_FINISHED	1307
7.301.2.12SOLVER_COL_TO_EQUATIONS_SETS_MAP	1307
7.301.2.13SOLVER_ROW_TO_EQUATIONS_SET_MAPS	1307
7.302TYPES::SOLUTION_PTR_TYPE Struct Reference	1309
7.302.1 Detailed Description	1309
7.302.2 Member Data Documentation	1309
7.302.2.1 PTR	1309
7.303TYPES::SOLUTION_TYPE Struct Reference	1310
7.303.1 Detailed Description	1310

7.303.2 Member Data Documentation	1310
7.303.2.1 EQUATIONS_SET_ADDED_INDEX	1310
7.303.2.2 EQUATIONS_SET_TO_ADD	1310
7.303.2.3 LINEARITY	1311
7.303.2.4 PROBLEM	1311
7.303.2.5 SOLUTION_FINISHED	1311
7.303.2.6 SOLUTION_MAPPING	1311
7.303.2.7 SOLUTION_NUMBER	1311
7.303.2.8 SOLVER	1311
7.304TYPES::SOLVER_COL_TO_EQUATIONS_MAP_TYPE Struct Reference	1312
7.304.1 Detailed Description	1312
7.304.2 Member Data Documentation	1312
7.304.2.1 COUPLING_COEFFICIENTS	1312
7.304.2.2 EQUATIONS_COL_NUMBERS	1312
7.304.2.3 EQUATIONS_MATRIX_NUMBERS	1313
7.304.2.4 JACOBIAN_COL_NUMBER	1313
7.304.2.5 JACOBIAN_COUPLING_COEFFICIENT	1313
7.304.2.6 NUMBER_OF_LINEAR_EQUATIONS_MATRICES	1313
7.305TYPES::SOLVER_COL_TO_EQUATIONS_SET_MAP_TYPE Struct Reference	1314
7.305.1 Detailed Description	1314
7.305.2 Member Data Documentation	1314
7.305.2.1 EQUATIONS	1314
7.305.2.2 SOLVER_COL_TO_EQUATIONS_MAPS	1314
7.306TYPES::SOLVER_COL_TO_EQUATIONS_SETS_MAP_TYPE Struct Reference	1315
7.306.1 Detailed Description	1315
7.306.2 Member Data Documentation	1316
7.306.2.1 COLUMN_DOFS_MAPPING	1316
7.306.2.2 NUMBER_OF_COLUMNS	1316
7.306.2.3 NUMBER_OF_DOFS	1316
7.306.2.4 NUMBER_OF_GLOBAL_DOFS	1316
7.306.2.5 SOLUTION_MAPPING	1316
7.306.2.6 SOLVER_COL_TO_EQUATIONS_SET_MAPS	1316
7.306.2.7 SOLVER_DOF_TO_VARIABLE_MAPS	1316
7.306.2.8 SOLVER_MATRIX	1317
7.306.2.9 SOLVER_MATRIX_NUMBER	1317
7.306.2.10TOTAL_NUMBER_OF_DOFS	1317

7.307TYPES::SOLVER_DOF_TO_VARIABLE_MAP_TYPE Struct Reference	1318
7.307.1 Detailed Description	1318
7.307.2 Member Data Documentation	1318
7.307.2.1 ADDITIVE_CONSTANT	1318
7.307.2.2 EQUATIONS_SET_INDICES	1318
7.307.2.3 NUMBER_OF_EQUATIONS_SETS	1319
7.307.2.4 VARIABLE	1319
7.307.2.5 VARIABLE_COEFFICIENT	1319
7.307.2.6 VARIABLE_DOF	1319
7.308TYPES::SOLVER_JACOBIAN_TYPE Struct Reference	1320
7.308.1 Detailed Description	1320
7.308.2 Member Data Documentation	1320
7.308.2.1 JACOBIAN	1320
7.308.2.2 NUMBER_OF_COLUMNS	1320
7.308.2.3 SOLVER_MATRICES	1320
7.308.2.4 STORAGE_TYPE	1321
7.308.2.5 UPDATE_JACOBIAN	1321
7.309TYPES::SOLVER_MATRICES_TYPE Struct Reference	1322
7.309.1 Detailed Description	1322
7.309.2 Member Data Documentation	1323
7.309.2.1 LIBRARY_TYPE	1323
7.309.2.2 MATRICES	1323
7.309.2.3 NUMBER_OF_GLOBAL_ROWS	1323
7.309.2.4 NUMBER_OF_MATRICES	1323
7.309.2.5 NUMBER_OF_ROWS	1323
7.309.2.6 RESIDUAL	1323
7.309.2.7 RHS_VECTOR	1323
7.309.2.8 SOLUTION_MAPPING	1323
7.309.2.9 SOLVER	1324
7.309.2.10 SOLVER_MATRICES_FINISHED	1324
7.309.2.11 UPDATE_RESIDUAL	1324
7.309.2.12 UPDATE_RHS_VECTOR	1324
7.310TYPES::SOLVER_MATRIX_PTR_TYPE Struct Reference	1325
7.310.1 Detailed Description	1325
7.310.2 Member Data Documentation	1325
7.310.2.1 PTR	1325

7.311TYPES::SOLVER_MATRIX_TYPE Struct Reference	1326
7.311.1 Detailed Description	1326
7.311.2 Member Data Documentation	1326
7.311.2.1 MATRIX	1326
7.311.2.2 MATRIX_NUMBER	1326
7.311.2.3 NUMBER_OF_COLUMNS	1327
7.311.2.4 SOLVER_MATRICES	1327
7.311.2.5 SOLVER_VECTOR	1327
7.311.2.6 STORAGE_TYPE	1327
7.311.2.7 UPDATE_MATRIX	1327
7.312TYPES::SOLVER_ROW_TO_EQUATIONS_SET_MAP_TYPE Struct Reference	1328
7.312.1 Detailed Description	1328
7.312.2 Member Data Documentation	1328
7.312.2.1 COUPLING_COEFFICIENTS	1328
7.312.2.2 EQUATIONS_ROW_NUMBER	1328
7.312.2.3 EQUATIONS_SET	1328
7.312.2.4 NUMBER_OF_ROWS	1329
7.313TYPES::SOLVER_TYPE Struct Reference	1330
7.313.1 Detailed Description	1330
7.313.2 Member Data Documentation	1331
7.313.2.1 EIGENPROBLEM_SOLVER	1331
7.313.2.2 LINEAR_SOLVER	1331
7.313.2.3 NONLINEAR_SOLVER	1331
7.313.2.4 OUTPUT_TYPE	1331
7.313.2.5 SOLUTION	1331
7.313.2.6 SOLUTION_MAPPING	1331
7.313.2.7 SOLVE_TYPE	1331
7.313.2.8 SOLVER_FINISHED	1332
7.313.2.9 SOLVER_MATRICES	1332
7.313.2.10SPARSITY_TYPE	1332
7.313.2.11TIME_INTEGRATION_SOLVER	1332
7.314TYPES::TIME_INTEGRATION_SOLVER_TYPE Struct Reference	1333
7.314.1 Detailed Description	1333
7.314.2 Member Data Documentation	1333
7.314.2.1 SOLVER	1333
7.314.2.2 SOLVER_LIBRARY	1333

7.315TYPES::VARIABLE_TO_EQUATIONS_COLUMN_MAP_TYPE Struct Reference	1334
7.315.1 Detailed Description	1334
7.315.2 Member Data Documentation	1334
7.315.2.1 COLUMN_DOF	1334
7.316TYPES::VARIABLE_TO_EQUATIONS_JACOBIAN_MAP_TYPE Struct Reference	1335
7.316.1 Detailed Description	1335
7.316.2 Member Data Documentation	1335
7.316.2.1 DOF_TO_COLUMNS_MAP	1335
7.316.2.2 DOF_TO_ROWS_MAP	1335
7.316.2.3 VARIABLE	1335
7.316.2.4 VARIABLE_TYPE	1336
7.317TYPES::VARIABLE_TO_EQUATIONS_MATRICES_MAP_TYPE Struct Reference	1337
7.317.1 Detailed Description	1337
7.317.2 Member Data Documentation	1337
7.317.2.1 DOF_TO_COLUMNS_MAPS	1337
7.317.2.2 DOF_TO_ROWS_MAP	1338
7.317.2.3 EQUATIONS_MATRIX_NUMBERS	1338
7.317.2.4 NUMBER_OF_LINEAR_EQUATIONS_MATRICES	1338
7.317.2.5 VARIABLE	1338
7.317.2.6 VARIABLE_INDEX	1338
7.317.2.7 VARIABLE_TYPE	1338
7.318TYPES::VARIABLE_TO_SOLVER_COL_MAP_TYPE Struct Reference	1339
7.318.1 Detailed Description	1339
7.318.2 Member Data Documentation	1339
7.318.2.1 ADDITIVE_CONSTANTS	1339
7.318.2.2 COLUMN_NUMBERS	1339
7.318.2.3 COUPLING_COEFFICIENTS	1340
7.319TYPES::VECTOR_TYPE Struct Reference	1341
7.319.1 Detailed Description	1341
7.319.2 Member Data Documentation	1341
7.319.2.1 DATA_DP	1341
7.319.2.2 DATA_INTG	1342
7.319.2.3 DATA_L	1342
7.319.2.4 DATA_SP	1342
7.319.2.5 DATA_TYPE	1342
7.319.2.6 ID	1342

7.319.2.7 N	1342
7.319.2.8 SIZE	1342
7.319.2.9 VECTOR_FINISHED	1342
8 File Documentation	1345
8.1 additional_doc/Installation.dox File Reference	1345
8.2 additional_doc/library_commands.dox File Reference	1346
8.2.1 Detailed Description	1346
8.3 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/analytic_analysis_routines.f90 File Reference	1347
8.3.1 Detailed Description	1348
8.3.2 LICENSE	1349
8.4 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/base_routines.f90 File Reference	1350
8.4.1 Detailed Description	1355
8.4.2 LICENSE	1355
8.5 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/basis_routines.f90 File Reference	1356
8.6 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/binary_file_c.c File Reference	1357
8.6.1 Define Documentation	1357
8.6.1.1 CHARTYPE	1357
8.6.1.2 COMPLEXTYPE	1358
8.6.1.3 DOUBLECOMPLEXTYPE	1358
8.6.1.4 DOUBLETYPE	1358
8.6.1.5 FLIPENDIAN	1358
8.6.1.6 FLOATTYPE	1358
8.6.1.7 INTEGERTYPE	1358
8.6.1.8 LOGICALTYPE	1358
8.6.1.9 LONGINTTYPE	1358
8.6.1.10 MAXBINFILES	1358
8.6.1.11 QUADRUPLECOMPLEXTYPE	1358
8.6.1.12 QUADRUPLETYP	1359
8.6.1.13 SAMEENDIAN	1359
8.6.1.14 SHORTINTTYPE	1359
8.6.2 Typedef Documentation	1359
8.6.2.1 logical	1359
8.6.3 Function Documentation	1359
8.6.3.1 BinaryCloseFile	1359
8.6.3.2 BinaryOpenFile	1359

8.6.3.3	BinaryReadFile	1359
8.6.3.4	BinarySetFile	1359
8.6.3.5	BinarySkipFile	1360
8.6.3.6	BinaryWriteFile	1360
8.6.3.7	IsBinaryFileOpen	1360
8.6.3.8	IsEndBinaryFile	1360
8.6.4	Variable Documentation	1360
8.6.4.1	binaryfiles	1360
8.7	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/binary_file_f.f90 File Reference	1361
8.7.1	Detailed Description	1363
8.7.2	LICENSE	1363
8.8	d:/Users/tyu011/workspace/OpenCMISS-trunk/srcblas.f90 File Reference	1365
8.8.1	Detailed Description	1365
8.8.2	LICENSE	1365
8.9	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/classical_field_routines.f90 File Reference	1367
8.9.1	Detailed Description	1368
8.9.2	LICENSE	1368
8.10	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/cmiss.f90 File Reference	1369
8.10.1	Detailed Description	1369
8.11	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/cmiss_mpi.f90 File Reference	1370
8.11.1	Detailed Description	1370
8.11.2	LICENSE	1370
8.12	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/cmiss_parmetis.f90 File Reference	1372
8.12.1	Detailed Description	1372
8.12.2	LICENSE	1372
8.13	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/cmiss_petsc.f90 File Reference	1374
8.13.1	Detailed Description	1381
8.13.2	LICENSE	1382
8.14	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/cmiss_petsc_types.f90 File Reference	1383
8.14.1	Detailed Description	1383
8.14.2	LICENSE	1383
8.15	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/computational_environment.f90 File Reference	1385
8.15.1	Detailed Description	1386
8.15.2	LICENSE	1386
8.16	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/constants.f90 File Reference	1388

8.17 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/coordinate_routines.f90 File Reference	1389
8.17.1 Detailed Description	1392
8.17.2 LICENSE	1392
8.18 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/distributed_matrix_vector.f90 File Reference	1394
8.19 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/domain_mappings.f90 File Reference	1395
8.19.1 Detailed Description	1396
8.19.2 LICENSE	1396
8.20 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/elasticity_routines.f90 File Reference	1397
8.20.1 Detailed Description	1397
8.20.2 LICENSE	1398
8.21 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/electromechanics_routines.f90 File Reference	1399
8.21.1 Detailed Description	1399
8.21.2 LICENSE	1399
8.22 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/equations_mapping_routines.f90 File Reference	1400
8.22.1 Detailed Description	1403
8.22.2 LICENSE	1403
8.23 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/equations_matrices_routines.f90 File Reference	1404
8.24 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/equations_set_constants.f90 File Reference	1405
8.24.1 Detailed Description	1409
8.24.2 LICENSE	1409
8.25 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/equations_set_routines.f90 File Reference	1411
8.26 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/f90c_c.c File Reference	1412
8.26.1 Typedef Documentation	1412
8.26.1.1 integer	1412
8.26.1.2 logical	1412
8.26.2 Function Documentation	1412
8.26.2.1 CStringLen	1412
8.26.2.2 PackCharacters	1412
8.26.2.3 UnPackCharacters	1412
8.27 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/f90c_f.f90 File Reference	1413
8.27.1 Detailed Description	1413
8.27.2 LICENSE	1413

8.28 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/field_IO_routines.f90 File Reference	1415
8.28.1 Detailed Description	1419
8.28.2 LICENSE	1419
8.29 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/field_routines.f90 File Reference	1420
8.29.1 Detailed Description	1425
8.29.2 LICENSE	1425
8.30 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/finite_elasticity_routines.f90 File Reference	1427
8.30.1 Detailed Description	1428
8.30.2 LICENSE	1428
8.31 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/finite_element_routines.f90 File Reference	1429
8.31.1 Detailed Description	1429
8.31.2 LICENSE	1429
8.32 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/fluid_mechanics_routines.f90 File Reference	1431
8.32.1 Detailed Description	1431
8.32.2 LICENSE	1431
8.33 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/generated_mesh_routines.f90 File Reference	1432
8.33.1 Detailed Description	1435
8.33.2 LICENSE	1435
8.34 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/input_output.f90 File Reference	1436
8.34.1 Detailed Description	1450
8.34.2 LICENSE	1450
8.35 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/iso_varying_string.f90 File Reference	1451
8.35.1 Detailed Description	1454
8.36 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/kinds.f90 File Reference	1455
8.36.1 Detailed Description	1456
8.36.2 LICENSE	1456
8.37 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/lapack.f90 File Reference	1457
8.37.1 Detailed Description	1457
8.37.2 LICENSE	1457
8.38 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/Laplace_equations_routines.f90 File Reference	1459
8.38.1 Detailed Description	1460
8.38.2 LICENSE	1460

8.39 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/linear_elasticity_routines.f90 File Reference	1462
8.39.1 Detailed Description	1462
8.39.2 LICENSE	1462
8.40 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/lists.f90 File Reference	1464
8.40.1 Detailed Description	1468
8.40.2 LICENSE	1468
8.41 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine_constants_aix.f90 File Reference	1469
8.41.1 Detailed Description	1469
8.41.2 LICENSE	1470
8.42 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine_constants_irix.f90 File Reference	1471
8.42.1 Detailed Description	1471
8.42.2 LICENSE	1471
8.43 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine_constants_linux.f90 File Reference	1472
8.43.1 Detailed Description	1472
8.43.2 LICENSE	1472
8.44 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine_constants_vms.f90 File Reference	1473
8.44.1 Detailed Description	1473
8.44.2 LICENSE	1473
8.45 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine_constants_win32.f90 File Reference	1474
8.45.1 Detailed Description	1474
8.45.2 LICENSE	1474
8.46 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/math.f90 File Reference	1475
8.46.1 Detailed Description	1476
8.46.2 LICENSE	1476
8.47 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/matrix_vector.f90 File Reference	1478
8.47.1 Detailed Description	1487
8.47.2 LICENSE	1487
8.47.3 MATRIX STORAGE STRUCTURES	1488
8.47.3.1 COMPRESSED-ROW STORAGE:	1488
8.47.3.2 COMPRESSED-COLUMN STORAGE:	1488
8.47.3.3 ROW-COLUMN STORAGE:	1489
8.48 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/mesh_routines.f90 File Reference	1490

8.48.1	Detailed Description	1496
8.48.2	LICENSE	1496
8.49	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/node_routines.f90 File Reference	1498
8.49.1	Detailed Description	1498
8.49.2	LICENSE	1498
8.50	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/problem_constants.f90 File Reference	1500
8.50.1	Detailed Description	1502
8.50.2	LICENSE	1502
8.51	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/problem_routines.f90 File Reference	1504
8.51.1	Detailed Description	1508
8.51.2	LICENSE	1508
8.51.3	Function Documentation	1508
8.51.3.1	PROBLEM_SOLUTION_JACOBIAN_EVALUATE_PETSC	1508
8.51.3.2	PROBLEM_SOLUTION_RESIDUAL_EVALUATE_PETSC	1509
8.52	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/region_routines.f90 File Reference	1510
8.52.1	Detailed Description	1511
8.52.2	LICENSE	1511
8.53	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/solution_mapping_routines.f90 File Reference	1512
8.54	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/solver_matrices_routines.f90 File Reference	1513
8.54.1	Detailed Description	1514
8.54.2	LICENSE	1514
8.55	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/solver_routines.f90 File Reference	1516
8.55.1	Detailed Description	1524
8.55.2	LICENSE	1524
8.56	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/sorting.f90 File Reference	1526
8.56.1	Detailed Description	1526
8.56.2	LICENSE	1526
8.57	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/strings.f90 File Reference	1528
8.57.1	Detailed Description	1532
8.57.2	LICENSE	1532
8.58	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/timer_c.c File Reference	1534
8.58.1	Define Documentation	1534
8.58.1.1	SYSTEM_CPU	1534
8.58.1.2	TOTAL_CPU	1534
8.58.1.3	USER_CPU	1534

8.58.2 Function Documentation	1534
8.58.2.1 CPUTimer	1534
8.59 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/timer_f.f90 File Reference	1535
8.59.1 Detailed Description	1535
8.59.2 LICENSE	1535
8.60 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/trees.f90 File Reference	1537
8.60.1 Detailed Description	1539
8.60.2 LICENSE	1539
8.61 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/types.f90 File Reference	1540
8.61.1 Detailed Description	1548
8.61.2 LICENSE	1548

Chapter 1

openCMISS Documentation

An open source interactive computer program for Continuum Mechanics, Image analysis, Signal processing and System Identification. Target usage: Bioengineering application of finite element analysis, boundary element and collocation techniques.

1.1 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Chapter 2

Obtaining the Code and Setting up the Development Environment

2.1 Obtaining the Code and Libraries

To obtain the openCMISS source you need to check it out from the subversion repository. There are two parts to openCMISS to obtain - openCMISS itself and the various libraries it needs. In your root openCMISS directory, make the opencmiss and opencmissextras directories.

2.1.1 Obtain the Code

The openCMISS repository is at <https://opencmiss.svn.sourceforge.net/svnroot/opencmiss/cm>
To check out the main trunk of openCMISS issue the following command in the opencmiss directory:

```
svn co https://opencmiss.svn.sourceforge.net/svnroot/opencmiss/cm/trunk cm
```

If you are not familiar with subversion, have a look at <http://svnbook.red-bean.com>.

2.1.2 Obtain the Libraries

The openCMISS libraries repository is at <http://www.physiome.ox.ac.uk/svn/opencmissextras/cm>
To check out the main trunk of the various libraries required with openCMISS issue the following command in the opencmissextras directory:

```
svn co http://www.physiome.ox.ac.uk/svn/opencmissextras/cm/trunk/external/architecture cm/external/architecture
```

where architecture is the appropriate architecture for the machine. Possible architectures are:

- i386-win32
- i386-win32-debug
- i686-linux
- i686-linux-debug
- x86_64-linux
- x86_64-linux-debug
- rs6000-32-aix
- rs6000-32-aix-debug

Currently, the svn repository for openCMISS libraries is down. An alternative location for the libraries within the ABI is on hpc. Go to \bioengsmb and copy the necessary files. The folder structure is the same as svn repository.

2.1.3 Makefile Structure

The top level makefile will eventually build a library. In the examples directory there are separate compilable "applications" with individual makefiles. However, the library stuff isn't there as we need to code the bindings.

2.2 Programmer documentation

This programmer documentation is written using DocBook with the source located in the opencmiss/cm/doc/programmer_documentation/ folder. The source XML code can be transformed into either chunked or combined documents in PDF or HTML format. The Makefile can be used to perform the transformation using make single or make chunk in the above folder. PDFs can be generated using make fo Doxygen can be included .

2.3 Project Setup

2.3.1 On AIX 5.3 (HPC)

2.3.1.1 Set environment

Set environment variable to point to openCMISS

```
setenv OPENCMISS_ROOT <path to your opencmiss folder>
```

or

```
export OPENCMISS_ROOT=<path to your opencmiss folder>
```

Set environment variable to point to openCMISS-extras

```
setenv OPENCMISSEXTRAS_ROOT <path to your opencmissextras folder>
```

or

```
export OPENCMISSEXTRAS_ROOT=<path to your opencmissextras folder>
```

2.3.1.2 Set MPI

Create a file called hostfile.list in your home directory. Inside the file, add several lines of "hpc.bioeng.auckland.ac.nz" In .rhost file in the home direcotry, add "hpc <username>"

2.3.1.3 Compile

Change directory to opencmiss/cm Change directory to examples/ Use gmake. This should result in a binary that you can run in the bin/rs6000-32-aix folder.

2.3.2 On Ubuntu 8.04

2.3.2.1 Set environment

Set environment variable to point to opencmiss

```
setenv OPENCMISS_ROOT <path to your opencmiss folder>
```

or

```
export OPENCMISS_ROOT=<path to your opencmiss folder>
```

Set environment variable to point to openCMISS-extras

```
setenv OPENCMISS_EXTRAS_ROOT <path to your opencmissextras folder>
```

or

```
export OPENCMISS_EXTRAS_ROOT=<path to your opencmissextras folder>
```

It is also helpful to add the following

```
setenv PATH ${OPENCMISS_EXTRAS_ROOT}/cm/external/${archname}/bin:${PATH}
setenv PATH ${OPENCMISS_ROOT}/cm/bin/${archname}:${PATH}
```

where \${archname} is the appropriate architecture e.g., i686-linux, x86_64-linux. If you are using totalview you will also need to add

```
setenv LM_LICENSE_FILES <path-to-the-flex-directory>
```

2.3.2.2 Install Compilers

Download Intel Fotran Compiler from here. Extract the file and follow the install.htm to install

2.3.2.3 Compile

To build an example project:

```
make
```

To run the example project:

```
mpd & mpirun -n 2 path/to/the/execution/file
```

To debug the project using TotalView:

```
mpd & mpirun -tv 2 path/to/the/execution/file
```

2.3.3 On Windows XP (Visual Studio 2005)

2.3.3.1 Install Compilers

Download Intel Fortran Compiler from here. Execute the exe file and follow the installation wizard.

2.3.3.2 Install MPI

Download MPICH2 from here. You can either download the source archive and follow the README.windows file to install or download the installer to install. Set bin folder to the path To start the MPI, run

```
smpd -start
```

in command window. NOTE: as from MPICH2 version 1.0.7 the library names have changed. libmpich2 has now become libmpi!

2.3.3.3 Compile and Debug

Build the Fortran project under the debug mode and generate the opencmissstest-debug.exe file. In the C Project (since the Fortran projects do not support MPI cluster debugger), configure the debugging properties according to this. The MPIShim location is in the path similar to C: Files Visual Studio 8 Debugger\x86\mpishim.exe. Debug the C project.

2.3.3.4 Run

To run the project in the command window:

```
mpiexec -n 2 -localroot <path to the execution file>
```

2.3.4 On Windows Vista (Visual Studio 2008)

Install Compilers Download Intel Fortran Compiler from here. Execute the exe file and follow the installation wizard.

2.3.4.1

Before you install MPICH2 under Vista you must turn off User Account Control

1. Goto Start -> Control Panel
2. Double-click on User Accounts
3. Click "Turn User Account Control on or off"
4. Untick "Use User Account Control (UAC) to help protect your computer" and click OK
5. Restart your computer.

Download from here. Choose the Win32 IA32 (binary) option. Run the downloaded .msi file. Follow all instructions and install "For everybody". Once you have installed MPICH2 you can turn User Account Control back on. Follow the instructions above and in 4. tick the "Use User Account Control ...". NOTE: as from MPICH2 version 1.0.7 the library names have changed. libmpich2 has now become libmpi!

2.3.4.2 Compile

For each example, go into the VisualopenCMISS_08 folder. Double click the VisualopenCMISS project solution file to lauch Visual Studio.

2.4 Libraries Build (Optional)

2.4.1 Compiling PETSc

Note this is assuming you have the Intel Fortran compiler version 10.1.024. Adjust the version string as necessary.

2.4.1.1 Step1: Linux Environment installation and Compiler Environment Set up (For Windows only)

Under windows system:

- Install Cygwin if you need to. Cywin can be found here. Make sure you include the make and python modules when you install.
- Launch a Command Prompt Window
- Run the ifortvars.bat batch file to setup your Intel Fortran environment. e.g., "C:\Program Files\Intel\Compiler\Fortran\10.1.024\IA32\Bin\ifortvars.bat"
- Run the Cygwin batch file to setup the unix environment e.g., "C:\Cygwin\Cygwin.bat"

N.B: PetSc uses X, so make sure in linux environment, libX11-dev package is installed. Also make sure blas and lapack (-dev) packages are installed.

2.4.1.2 Step2: Compile PETSc

For Windows

- Change to the opencmissextras PETSc directory e.g., if opencmissextras root is E: and we are compiling PETSC version petsc-2.3.3-p8 then "cd /cygwin/e/opencmissextras/cm/external/packages/PETSc/petsc-2.3.3-p8"
- If you have MPICH2 version 1.0.7 or greater edit the python/BuildSystem/config/packages/MPI.py file. Find the self.liblist_mpich line. After the line "[fmpich2.lib', 'mpich2.lib'],", add the line "[fmpich2.lib', 'mpi.lib'],".
- PETSC_DIR=/cygdrive/e/opencmissextras/cm/external/packages/PETSc/petsc-2.3.3-p8; export PETSC_DIR
- For a debug install issue the following commands

```
PETSC_ARCH=cygwin-c-debug; export PETSC_ARCH
config/configure.py --prefix=/cygdrive/e/opencmissextras/cm/external/i386-win32-debug --with-shared=no
PETSC_ARCH=cygwin-c-debug; export PETSC_ARCH
```

For a non-debug install issue the following commands

```
PETSC_ARCH=cygwin-c-opt; export PETSC_ARCH
config/configure.py --prefix=/cygdrive/e/opencmissextras/cm/external/i386-win32 --with-shared=no --with-pic
PETSC_ARCH=cygwin-c-opt; export PETSC_ARCH
```

- make -e all
- make -e install

For AIX5.3

- Change to the opencmissextras PETSc directory. e.g /people/tyu011/workspace/opencmissextras/cm/external/packages/PETSc/2.3.3-p8
- setenv PETSC_DIR /people/tyu011/workspace/opencmissextras/cm/external/packages/PETSc/petsc-2.3.3-p8
- In config directory, copy the file aix5.1.0.0.py and rename it to aix5.3.0.0.py
- For a debug install issue the following commands

```
setenv PETSC_ARCH aix5.3.0.0
config/aix5.3.0.0.py --prefix=/people/tyu011/workspace/opencmissextras/cm/external/rs6000-32-aix-debug
setenv PETSC_ARCH aix5.3.0.0
```

For a non-debug install issue the following commands

```
setenv PETSC_ARCH aix5.3.0.0
config/aix5.3.0.0.py --prefix=/people/tyu011/workspace/opencmissextras/cm/external/rs6000-32-aix --without-debug
setenv PETSC_ARCH aix5.3.0.0
```

- make -e all
- make -e install

Chapter 3

Library Commands

3.1 OpenCMISS library structure

- Top Level - cmiss
- Basis Routines
- Coordinate System Routines
- Region Routines
 - Node Routines
 - Mesh Routines
 - Field Routines
 - Equations Set Routines
- Problems Routines
 - Solver Routines

3.2 Top Level - cmiss

To initialise **CMISS**, use:

SUBROUTINE **CMISS_INITIALISE** (ERR,ERROR,*)

Description: Initialises **CMISS**.

To finalise **CMISS**, use:

SUBROUTINE **CMISS_FINALISE** (ERR,ERROR,*)

Description: Finalises **CMISS**.

3.3 Basis Routines

3.4 Coordinate System Routines

3.5 Region Routines

3.5.1 Node Routines

3.5.2 Mesh Routines

3.5.3 Field Routines

3.5.4 Equations Set Routines

3.6 Problems Routines

3.6.1 Solver Routines

Chapter 4

Todo List

Member TYPES::BASIS_TYPE::NUMBER_OF_NODES_XI CHANGE TO NUMBER_OF_NODES_XIC i.e., xi coordinate index not xi???

Class TYPES::COORDINATE_SYSTEM_TYPE Have a list of coordinate systems and have a pointer in the coordinate_system_type back to the regions that use them.

Member TYPES::DECOMPOSITION_ELEMENTS_TYPE::ELEMENTS Change this to allocatable???

Member TYPES::DECOMPOSITION_TYPE::DOMAIN Change this to allocatable???

Member TYPES::DOMAIN_ELEMENTS_TYPE::ELEMENTS Change this to allocatable???

Member TYPES::DOMAIN_FACE_TYPE::XI_DIRECTION1 move this to the decomposition face type

Member TYPES::DOMAIN_FACE_TYPE::XI_DIRECTION2 move this to the decomposition face type

Member TYPES::DOMAIN_NODE_TYPE::SURROUNDING_ELEMENTS Change this to allocatable.

Member TYPES::DOMAIN_NODES_TYPE::NODES Change this to allocatable???

Member TYPES::EQUATIONS_MATRICES_RHS_TYPE::VECTOR rename this RHS_VECTOR

Member TYPES::EQUATIONS_MATRICES_SOURCE_TYPE::VECTOR rename this SOURCE_VECTOR

Member TYPES::EQUATIONS_SET_TYPE::FIXED_CONDITIONS Change name to BOUNDARY_CONDITIONS???

Member TYPES::EQUATIONS_TYPE::OUTPUT_TYPE move to equations set???

Member TYPES::EQUATIONS_TYPE::SPARSITY_TYPE move to equations set???

Member TYPES::FIELD_PARAM_TO_DOF_MAP_TYPE::ELEMENT_PARAM2DOF_MAP
Allow for multiple element parameters per element.

Member TYPES::FIELD_PARAM_TO_DOF_MAP_TYPE::NODE_PARAM2DOF_MAP Don't allocate too much memory if there are different numbers of derivatives for different nodes.

Member TYPES::FIELD_SCALING_TYPE::SCALE_FACTORS Make scale factors nodally based for now. Will have to revert to element based and extended to be a matrix to allow for a global derivative to be mapped onto many different element derivatives at the points that closes meshes or for inconsistent xi directions

Member TYPES::MESH_ELEMENTS_TYPE::ELEMENTS Should this be allocatable.

Member TYPES::MESH_NODE_TYPE::SURROUNDING_ELEMENTS Change this to allocatable.

Member TYPES::MESH_NODES_TYPE::NODES Should this be allocatable???

Member TYPES::NODES_TYPE::NODES Should this be allocatable?

Class TYPES::QUADRATURE_SCHEME_TYPE Also evaluate the product of the basis functions at gauss points for speed???

Class TYPES::VARIABLE_TO_SOLVER_COL_MAP_TYPE rename solver col to be solver dof here

Member BASE_ROUTINES::BASE_ROUTINES_FINALISE(**ERR, ERROR,*)** Finish this routine and deallocate memory.

File binary_file_f.f90 Fix naming convention and update to current code standard.

Member FIELD_ROUTINES::FIELD_COMPONENT_MESH_COM(**USER_NUMBER, FIELD_VARIABLE_NUMB**

change variable number to variable type.

File generated_mesh_routines.f90 Move generated regular mesh from mesh routines and generalise.

File lists.f90 Fix up and have this module use the sorting module for sorts

Member MESH_ROUTINES::DOMAIN_MAPPINGS_NODES_FINALISE(**DOMAIN_MAPPINGS, ERR, ERROR,*)**

pass in the nodes mapping

Member MESH_ROUTINES::DOMAIN_MAPPINGS_NODES_INITIALISE(**DOMAIN_MAPPINGS, ERR, ERROR,**

finalise on error

Member **MESH_ROUTINES::DOMAIN_TOPOLOGY_DOFS_FINALISE(TOPOLOGY, ERR, ERROR,*)**
pass in the dofs topolgy

Member **MESH_ROUTINES::DOMAIN_TOPOLOGY_DOFS_INITIALISE(TOPOLOGY, ERR, ERROR,*)**
finalise on exit

Member **MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_FINALISE(TOPOLOGY, ERR, ERROR,*)**
pass in the domain elements

Member **MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_INITIALISE(TOPOLOGY, ERR, ERROR,*)**
finalise on error

Member **MESH_ROUTINES::DOMAIN_TOPOLOGY_FINALISE(DOMAIN, ERR, ERROR,*)**
pass in domain topology

Member **MESH_ROUTINES::DOMAIN_TOPOLOGY_INITIALISE(DOMAIN, ERR, ERROR,*)**
finalise on error

Member **MESH_ROUTINES::DOMAIN_TOPOLOGY_LINES_FINALISE(TOPOLOGY, ERR, ERROR,*)**
pass in domain lines

Member **MESH_ROUTINES::DOMAIN_TOPOLOGY_LINES_INITIALISE(TOPOLOGY, ERR, ERROR,*)**
finalise on error

Member **MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_FINALISE(TOPOLOGY, ERR, ERROR,*)**
pass in domain nodes

Member **MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_INITIALISE(TOPOLOGY, ERR, ERROR,*)**
finalise on error

Member **MESH_ROUTINES::MESH_CREATE_FINISH(REGION, MESH, ERR, ERROR,*)**
remove region

Member **MESH_ROUTINES::MESH_TOPOLOGY_DOFS_FINALISE(TOPOLOGY, ERR, ERROR,*)**
pass in dofs

Member **MESH_ROUTINES::MESH_TOPOLOGY_DOFS_INITIALISE(TOPOLOGY, ERR, ERROR,*)**
finalise on error

Member **MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_BASIS_SET(GLOBAL_NUMBER, ELEMENTS, E)**
use user number.

Member MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_BASIS_SET(GLOBAL_NUMBER, ELEMENTS, ELEM_BASIS, *ELEMENT_BASIS)
get method?

Member MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_DESTROY(TOPOLOGY, ERR, ERROR,*)
as this is a user routine it should take a mesh pointer like create start and finish? Split this into
destroy and finalise?

Member MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_BASIS_GET(GLOBAL_NUMBER, ELEMENT_BASIS, *ELEMENT_BASIS)
should take user number

Member MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_BASIS_SET(GLOBAL_NUMBER, ELEMENT_BASIS, *ELEMENT_BASIS)
should take user number

Member MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_NODES_GET(GLOBAL_NUMBER, ELEMENT_NODES, *ELEMENT_NODES)
specify by user number not global number

Member MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_NODES_SET(GLOBAL_NUMBER, ELEMENT_NODES, *ELEMENT_NODES)
specify by user number not global number

Member MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_FINALISE(TOPOLOGY, ERR, ERROR,*)
pass in elements

Member MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_INITIALISE(TOPOLOGY, ERR, ERROR,*)
finalise on error

Member MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_NUMBER_GET(GLOBAL_NUMBER, USER_NUMBER, *USER_NUMBER)
Check that the user number doesn't already exist.

Member MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_NUMBER_SET(GLOBAL_NUMBER, USER_NUMBER, *USER_NUMBER)
Check that the user number doesn't already exist.

Member MESH_ROUTINES::MESH_TOPOLOGY_FINALISE(MESH, ERR, ERROR,*) pass in
the mesh topology

Member MESH_ROUTINES::MESH_TOPOLOGY_INITIALISE(MESH, ERR, ERROR,*)
finalise on error

Member MESH_ROUTINES::MESH_TOPOLOGY_NODES_FINALISE(TOPOLOGY, ERR, ERROR,*)
pass in nodes

Member MESH_ROUTINES::MESH_TOPOLOGY_NODES_INITIALISE(TOPOLOGY, ERR, ERROR,*)
finalise on errors

Member MESH_ROUTINES::MESHES_FINALISE(REGION, ERR, ERROR,*) pass in meshes

Member MESH_ROUTINES::MESHES_INITIALISE(REGION, ERR, ERROR,*) finalise on error

Member PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_CREATE_START(PROBLEM, ERR, ERROR,*)
Should this return a pointer to the problem solution???

Member LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_CALCULATE(FIELD, EQUA
for regular mesh only

Member LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_INTE(FIELD, NODE_NUMBER, COMPO
this is more generic, move to other place.

Group FIELD_ROUTINES_VariableTypes sort out variable access routines so that you are always accessing by variable type rather than variable number.

Group FIELD_ROUTINES_ParameterSetTypes make program defined constants negative?

Chapter 5

Module Documentation

5.1 BASE_ROUTINES::OutputType

Output type parameter.

Variables

- INTEGER(INTG), parameter [BASE_ROUTINES::GENERAL_OUTPUT_TYPE = 1](#)
General output type.
- INTEGER(INTG), parameter [BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE = 2](#)
Diagnostic output type.
- INTEGER(INTG), parameter [BASE_ROUTINES::TIMING_OUTPUT_TYPE = 3](#)
Timing output type.
- INTEGER(INTG), parameter [BASE_ROUTINES::ERROR_OUTPUT_TYPE = 4](#)
Error output type.
- INTEGER(INTG), parameter [BASE_ROUTINES::HELP_OUTPUT_TYPE = 5](#)
Help output type.

5.1.1 Detailed Description

Output type parameter.

See also:

[BASE_ROUTINES](#)

5.1.2 Variable Documentation

5.1.2.1 INTEGER(INTG), parameter [BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE = 2](#)

Diagnostic output type.

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES](#)

Definition at line 64 of file base_routines.f90.

Referenced by COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_TYPE_INITIALISE(), COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_FINISH(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE(), MESH_ROUTINES::DECOMPOSITION_CREATE_FINISH(), MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE(), MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(), MESH_ROUTINES::DOMAIN_TOPOLOGY_INITIALISE_FROM_MESH(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_CALCULATE(), FIELD_ROUTINES::FIELD_CREATE_FINISH(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), MESH_ROUTINES::MESH_CREATE_FINISH(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_FINISH(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE(), NODE_ROUTINES::NODES_CREATE_FINISH(), PROBLEM_ROUTINES::PROBLEM_CREATE_FINISH(), REGION_ROUTINES::REGION_CREATE_FINISH(), REGION_ROUTINES::REGION_SUB_REGION_CREATE_FINISH(), and SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_CALCULATE().

5.1.2.2 INTEGER(INTG),parameter BASE_ROUTINES::ERROR_OUTPUT_TYPE = 4

Error output type.

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES](#)

Definition at line 66 of file base_routines.f90.

5.1.2.3 INTEGER(INTG),parameter BASE_ROUTINES::GENERAL_OUTPUT_TYPE = 1

General output type.

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES](#)

Definition at line 63 of file base_routines.f90.

Referenced by BINARY_FILE::OPEN_CMISS_BINARY_FILE(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE(), SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE(), SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESearch_SOLVE(), and SOLVER_ROUTINES::SOLVER_SOLVE().

5.1.2.4 INTEGER(INTG),parameter BASE_ROUTINES::HELP_OUTPUT_TYPE = 5

Help output type.

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES](#)

Definition at line 67 of file base_routines.f90.

5.1.2.5 INTEGER(INTG),parameter BASE_ROUTINES::TIMING_OUTPUT_TYPE = 3

Timing output type.

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES](#)

Definition at line 65 of file base_routines.f90.

5.2 BASE_ROUTINES::FileUnits

File unit parameters.

Variables

- INTEGER(INTG), parameter **BASE_ROUTINES::ECHO_FILE_UNIT** = 10

File unit for echo files.

- INTEGER(INTG), parameter **BASE_ROUTINES::DIAGNOSTICS_FILE_UNIT** = 11

File unit for diagnostic files.

- INTEGER(INTG), parameter **BASE_ROUTINES::TIMING_FILE_UNIT** = 12

File unit for timing files.

- INTEGER(INTG), parameter **BASE_ROUTINES::LEARN_FILE_UNIT** = 13

File unit for learn files.

- INTEGER(INTG), parameter **BASE_ROUTINES::IO1_FILE_UNIT** = 21

File unit for general IO 1 files.

- INTEGER(INTG), parameter **BASE_ROUTINES::IO2_FILE_UNIT** = 22

File unit for general IO 2 files.

- INTEGER(INTG), parameter **BASE_ROUTINES::IO3_FILE_UNIT** = 23

File unit for general IO 3 files.

- INTEGER(INTG), parameter **BASE_ROUTINES::IO4_FILE_UNIT** = 24

File unit for general IO 4 files.

- INTEGER(INTG), parameter **BASE_ROUTINES::IO5_FILE_UNIT** = 25

File unit for general IO 5 files.

- INTEGER(INTG), parameter **BASE_ROUTINES::TEMPORARY_FILE_UNIT** = 80

File unit for temporary files.

- INTEGER(INTG), parameter **BASE_ROUTINES::OPEN_COMFILE_UNIT** = 90

File unit for open command files.

- INTEGER(INTG), parameter **BASE_ROUTINES::START_READ_COMFILE_UNIT** = 90

First file unit for read command files.

- INTEGER(INTG), parameter **BASE_ROUTINES::STOP_READ_COMFILE_UNIT** = 99

Last file unit for read command files.

5.2.1 Detailed Description

File unit parameters.

See also:

[BASE_ROUTINES](#)

5.2.2 Variable Documentation

5.2.2.1 INTEGER(INTG),parameter BASE_ROUTINES::DIAGNOSTICS_FILE_UNIT = 11

File unit for diagnostic files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 75 of file base_routines.f90.

5.2.2.2 INTEGER(INTG),parameter BASE_ROUTINES::ECHO_FILE_UNIT = 10

File unit for echo files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 74 of file base_routines.f90.

5.2.2.3 INTEGER(INTG),parameter BASE_ROUTINES::IO1_FILE_UNIT = 21

File unit for general IO 1 files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 78 of file base_routines.f90.

5.2.2.4 INTEGER(INTG),parameter BASE_ROUTINES::IO2_FILE_UNIT = 22

File unit for general IO 2 files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 79 of file base_routines.f90.

5.2.2.5 INTEGER(INTG),parameter BASE_ROUTINES::IO3_FILE_UNIT = 23

File unit for general IO 3 files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 80 of file base_routines.f90.

5.2.2.6 INTEGER(INTG),parameter BASE_ROUTINES::IO4_FILE_UNIT = 24

File unit for general IO 4 files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 81 of file base_routines.f90.

5.2.2.7 INTEGER(INTG),parameter BASE_ROUTINES::IO5_FILE_UNIT = 25

File unit for general IO 5 files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 82 of file base_routines.f90.

5.2.2.8 INTEGER(INTG),parameter BASE_ROUTINES::LEARN_FILE_UNIT = 13

File unit for learn files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 77 of file base_routines.f90.

5.2.2.9 INTEGER(INTG),parameter BASE_ROUTINES::OPEN_COMFILE_UNIT = 90

File unit for open command files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 84 of file base_routines.f90.

5.2.2.10 INTEGER(INTG),parameter BASE_ROUTINES::START_READ_COMFILE_UNIT = 90

First file unit for read command files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 85 of file base_routines.f90.

5.2.2.11 INTEGER(INTG),parameter BASE_ROUTINES::STOP_READ_COMFILE_UNIT = 99

Last file unit for read command files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 86 of file base_routines.f90.

5.2.2.12 INTEGER(INTG),parameter BASE_ROUTINES::TEMPORARY_FILE_UNIT = 80

File unit for temporary files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 83 of file base_routines.f90.

5.2.2.13 INTEGER(INTG),parameter BASE_ROUTINES::TIMING_FILE_UNIT = 12

File unit for timing files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 76 of file base_routines.f90.

5.3 BASE_ROUTINES::DiagnosticTypes

Diganostic type parameters.

Variables

- INTEGER(INTG), parameter [BASE_ROUTINES::ALL_DIAG_TYPE = 1](#)
Type for setting diagnostic output in all routines.
- INTEGER(INTG), parameter [BASE_ROUTINES::IN_DIAG_TYPE = 2](#)
Type for setting diagnostic output in one routine.
- INTEGER(INTG), parameter [BASE_ROUTINES::FROM_DIAG_TYPE = 3](#)
Type for setting diagnostic output from one routine downwards.

5.3.1 Detailed Description

Diganostic type parameters.

See also:

[BASE_ROUTINES](#)

5.3.2 Variable Documentation

5.3.2.1 INTEGER(INTG),parameter [BASE_ROUTINES::ALL_DIAG_TYPE = 1](#)

Type for setting diagnostic output in all routines.

See also:

[BASE_ROUTINES::DiagnosticTypes](#),[BASE_ROUTINES](#)

Definition at line 93 of file base_routines.f90.

5.3.2.2 INTEGER(INTG),parameter [BASE_ROUTINES::FROM_DIAG_TYPE = 3](#)

Type for setting diagnostic output from one routine downwards.

See also:

[BASE_ROUTINES::DiagnosticTypes](#),[BASE_ROUTINES](#)

Definition at line 95 of file base_routines.f90.

5.3.2.3 INTEGER(INTG),parameter BASE_ROUTINES::IN_DIAG_TYPE = 2

Type for setting diagnostic output in one routine.

See also:

[BASE_ROUTINES::DiagnosticTypes](#),[BASE_ROUTINES](#)

Definition at line 94 of file base_routines.f90.

5.4 BASE_ROUTINES::TimingTypes

Timing type parameters.

Variables

- INTEGER(INTG), parameter **BASE_ROUTINES::ALL_TIMING_TYPE = 1**
Type for setting timing output in all routines.
- INTEGER(INTG), parameter **BASE_ROUTINES::IN_TIMING_TYPE = 2**
Type for setting timing output in one routine.
- INTEGER(INTG), parameter **BASE_ROUTINES::FROM_TIMING_TYPE = 3**
Type for setting timing output from one routine downwards.

5.4.1 Detailed Description

Timing type parameters.

See also:

[BASE_ROUTINES](#)

5.4.2 Variable Documentation

5.4.2.1 INTEGER(INTG),parameter **BASE_ROUTINES::ALL_TIMING_TYPE = 1**

Type for setting timing output in all routines.

See also:

[BASE_ROUTINES::TimingTypes](#),[BASE_ROUTINES](#)

Definition at line 102 of file base_routines.f90.

5.4.2.2 INTEGER(INTG),parameter **BASE_ROUTINES::FROM_TIMING_TYPE = 3**

Type for setting timing output from one routine downwards.

See also:

[BASE_ROUTINES::TimingTypes](#),[BASE_ROUTINES](#)

Definition at line 104 of file base_routines.f90.

5.4.2.3 INTEGER(INTG),parameter BASE_ROUTINES::IN_TIMING_TYPE = 2

Type for setting timing output in one routine.

See also:

[BASE_ROUTINES::TimingTypes](#),[BASE_ROUTINES](#)

Definition at line 103 of file base_routines.f90.

5.5 COORDINATE_ROUTINES::CoordinateSystemTypes

Variables

- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_RECTANGULAR_CARTESIAN_TYPE = 1](#)
Rectangular Cartesian coordinate system type.
- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_CYCLINDRICAL_POLAR_TYPE = 2](#)
Cylindrical polar coordinate system type.
- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_SPHERICAL_POLAR_TYPE = 3](#)
Spherical polar coordinate system type.
- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_PROLATE_SPHEROIDAL_TYPE = 4](#)
Prolate spheroidal coordinate system type.
- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_OBLATE_SPHEROIDAL_TYPE = 5](#)
Oblate spheroidal coordinate system type.

5.5.1 Detailed Description

See also:

[COORDINATE_ROUTINES](#) Coordinate system type parameters

5.5.2 Variable Documentation

5.5.2.1 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_CYCLINDRICAL_POLAR_TYPE = 2

Cylindrical polar coordinate system type.

See also:

[COORDINATE_ROUTINES](#)_CoordinateSystemTypes, [COORDINATE_ROUTINES](#)

Definition at line 68 of file coordinate_routines.f90.

Referenced by [COORDINATE_ROUTINES::CO\(\)](#), [COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_DP\(\)](#), [COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_SP\(\)](#), [COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_DP\(\)](#), [COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_SP\(\)](#), [COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE_DP\(\)](#), [COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_NORM\(\)](#), [COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST\(\)](#), [COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_PARAMETERS_ADJUST\(\)](#), [COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE\(\)](#), [COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET_PTR\(\)](#), [COORDINATE_ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE\(\)](#),

COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_PTR(), COORDINATE_ROUTINES::D2ZX_DP(), COORDINATE_ROUTINES::DXZ_DP(), and COORDINATE_ROUTINES::DZX_DP().

5.5.2.2 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_OBLATE_SPHEROIDAL_TYPE = 5

Oblate spheroidal coordinate system type.

See also:

COORDINATE_ROUTINES_CoordinateSystemTypes, [COORDINATE_ROUTINES](#)

Definition at line 71 of file coordinate_routines.f90.

Referenced by COORDINATE_ROUTINES::CO(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_SP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_SP(), COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE_DP(), COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_NORM(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_PARAMETERS_ADJUST(), COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_GET(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_PTR(), COORDINATE_ROUTINES::D2ZX_DP(), COORDINATE_ROUTINES::DXZ_DP(), and COORDINATE_ROUTINES::DZX_DP().

5.5.2.3 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_PROLATE_SPHEROIDAL_TYPE = 4

Prolate spheroidal coordinate system type.

See also:

COORDINATE_ROUTINES_CoordinateSystemTypes, [COORDINATE_ROUTINES](#)

Definition at line 70 of file coordinate_routines.f90.

Referenced by COORDINATE_ROUTINES::CO(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_SP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_SP(), COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE_DP(), COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_NORM(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_PARAMETERS_ADJUST(), COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_GET(),

COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_PTR(), COORDINATE_ROUTINES::D2ZX_DP(), COORDINATE_ROUTINES::DXZ_DP(), and COORDINATE_ROUTINES::DZX_DP().

5.5.2.4 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_RECTANGULAR_CARTESIAN_TYPE = 1

Rectangular Cartesian coordinate system type.

See also:

[COORDINATE_ROUTINES_CoordinateSystemTypes](#), [COORDINATE_ROUTINES](#)

Definition at line 67 of file coordinate_routines.f90.

Referenced by COORDINATE_ROUTINES::CO(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_SP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_SP(), COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE_DP(), COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_NORM(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_PARAMETERS_ADJUST(), COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_START(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEMS_INITIALISE(), COORDINATE_ROUTINES::D2ZX_DP(), COORDINATE_ROUTINES::DXZ_DP(), COORDINATE_ROUTINES::DZX_DP(), FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET(), and GENERATED_MESH_ROUTINES::GENERATED_MESH_REGULAR_CREATE_FINISH().

5.5.2.5 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_SPHERICAL_POLAR_TYPE = 3

Spherical polar coordinate system type.

See also:

[COORDINATE_ROUTINES_CoordinateSystemTypes](#), [COORDINATE_ROUTINES](#)

Definition at line 69 of file coordinate_routines.f90.

Referenced by COORDINATE_ROUTINES::CO(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_SP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_SP(), COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE_DP(), COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_NORM(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_

PARAMETERS_ADJUST(), COORDINATE_ROUTINES::COORDINATE_METRICS_-
CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET_-
PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE(),
COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_-
PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_PTR(), COORDINATE_-
ROUTINES::D2ZX_DP(), COORDINATE_ROUTINES::DXZ_DP(), and COORDINATE_-
ROUTINES::DZX_DP().

5.6 COORDINATE_ROUTINES::RadialInterpolations

The type of radial interpolation for polar coordinate systems.

Variables

- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_NO_RADIAL_INTERPOLATION_TYPE = 0](#)
No radial interpolation.
- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_RADIAL_INTERPOLATION_TYPE = 1](#)
r radial interpolation
- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_RADIAL_SQUARED_INTERPOLATION_TYPE = 2](#)
 r^2 radial interpolation
- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_RADIAL_CUBED_INTERPOLATION_TYPE = 3](#)
 r^3 radial interpolation

5.6.1 Detailed Description

The type of radial interpolation for polar coordinate systems.

See also:

[COORDINATE_ROUTINES](#)

5.6.2 Variable Documentation

5.6.2.1 INTEGER(INTG),parameter [COORDINATE_ROUTINES::COORDINATE_NO_RADIAL_INTERPOLATION_TYPE = 0](#)

No radial interpolation.

See also:

[COORDINATE_ROUTINES::RadialInterpolations](#), [COORDINATE_ROUTINES](#)

Definition at line 78 of file coordinate_routines.f90.

Referenced by [COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_START\(\)](#), and [COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR\(\)](#).

**5.6.2.2 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_RADIAL_-
CUBED_INTERPOLATION_TYPE = 3**

r^3 radial interpolation

See also:

[COORDINATE_ROUTINES::RadialInterpolations](#), [COORDINATE_ROUTINES](#)

Definition at line 81 of file coordinate_routines.f90.

Referenced by COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_PARAMETERS_ADJUST(), and COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR().

**5.6.2.3 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_RADIAL_-
INTERPOLATION_TYPE = 1**

r radial interpolation

See also:

[COORDINATE_ROUTINES::RadialInterpolations](#), [COORDINATE_ROUTINES](#)

Definition at line 79 of file coordinate_routines.f90.

Referenced by COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_PARAMETERS_ADJUST(), and COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR().

**5.6.2.4 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_RADIAL_-
SQUARED_INTERPOLATION_TYPE = 2**

r^2 radial interpolation

See also:

[COORDINATE_ROUTINES::RadialInterpolations](#), [COORDINATE_ROUTINES](#)

Definition at line 80 of file coordinate_routines.f90.

Referenced by COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_PARAMETERS_ADJUST(), and COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR().

5.7 COORDINATE_ROUTINES::JacobianType

The type of Jacobian to return when coordinate metrics are calculated.

Variables

- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_JACOBIAN_LINE_TYPE = 1](#)
Line type Jacobian.
- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_JACOBIAN_AREA_TYPE = 2](#)
Area type Jacobian.
- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_JACOBIAN_VOLUME_TYPE = 3](#)
Volume type Jacobian.

5.7.1 Detailed Description

The type of Jacobian to return when coordinate metrics are calculated.

See also:

[COORDINATE_ROUTINES](#)

5.7.2 Variable Documentation

5.7.2.1 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_JACOBIAN_AREA_TYPE = 2

Area type Jacobian.

See also:

[COORDINATE_ROUTINES_JacobianTypes](#),[COORDINATE_ROUTINES](#)

Definition at line 89 of file coordinate_routines.f90.

Referenced by [COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE\(\)](#).

5.7.2.2 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_JACOBIAN_LINE_TYPE = 1

Line type Jacobian.

See also:

[COORDINATE_ROUTINES_JacobianTypes](#),[COORDINATE_ROUTINES](#)

Definition at line 88 of file coordinate_routines.f90.

Referenced by [COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE\(\)](#).

**5.7.2.3 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_-
JACOBIAN_VOLUME_TYPE = 3**

Volume type Jacobian.

See also:

COORDINATE_ROUTINES_JacobianTypes,[COORDINATE_ROUTINES](#)

Definition at line 90 of file coordinate_routines.f90.

Referenced by COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE().

5.8 DOMAIN_MAPPINGS::DomainType

Variables

- INTEGER(INTG), parameter [DOMAIN_MAPPINGS::DOMAIN_LOCAL_INTERNAL = 1](#)
The domain item is internal to the domain.
- INTEGER(INTG), parameter [DOMAIN_MAPPINGS::DOMAIN_LOCAL_BOUNDARY = 2](#)
The domain item is on the boundary of the domain.
- INTEGER(INTG), parameter [DOMAIN_MAPPINGS::DOMAIN_LOCAL_GHOST = 3](#)
The domain item is ghosted from another domain.

5.8.1 Detailed Description

See also:

[DOMAIN_MAPPINGS](#)

5.8.2 Variable Documentation

5.8.2.1 INTEGER(INTG),parameter DOMAIN_MAPPINGS::DOMAIN_LOCAL_BOUNDARY = 2

The domain item is on the boundary of the domain.

See also:

[DOMAIN_MAPPINGS::DomainType](#),[DOMAIN_MAPPINGS](#)

Definition at line 64 of file domain_mappings.f90.

Referenced by MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET().

5.8.2.2 INTEGER(INTG),parameter DOMAIN_MAPPINGS::DOMAIN_LOCAL_GHOST = 3

The domain item is ghosted from another domain.

See also:

[DOMAIN_MAPPINGS::DomainType](#),[DOMAIN_MAPPINGS](#)

Definition at line 65 of file domain_mappings.f90.

Referenced by MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(), and FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

5.8.2.3 INTEGER(INTG),parameter DOMAIN_MAPPINGS::DOMAIN_LOCAL_INTERNAL = 1

The domain item is internal to the domain.

See also:

[DOMAIN_MAPPINGS::DomainType](#), [DOMAIN_MAPPINGS](#)

Definition at line 63 of file domain_mappings.f90.

Referenced by [MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET\(\)](#), and [FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET\(\)](#).

5.9 EQUATIONS_SET_CONSTANTS::SetupTypes

Setup type parameters.

Variables

- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_INITIAL_TYPE = 1`
Initial setup.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_GEOMETRY_TYPE = 2`
Geometry setup.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_DEPENDENT_TYPE = 3`
Dependent variables.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_MATERIALS_TYPE = 4`
Materials setup.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_SOURCE_TYPE = 5`
Source setup.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_SOURCE_MATERIALS_TYPE = 6`
Source materials setup.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_ANALYTIC_TYPE = 7`
Analytic setup.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FIXED_CONDITIONS_TYPE = 8`
Fixed conditions.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_EQUATIONS_TYPE = 9`
Equations setup.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FINAL_TYPE = 9`
Final setup.

5.9.1 Detailed Description

Setup type parameters.

See also:

[EQUATIONS_SET_CONSTANTS](#)

5.9.2 Variable Documentation

5.9.2.1 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_- SETUP_ANALYTIC_TYPE = 7

Analytic setup.

See also:

[EQUATIONS_SET_CONSTANTS::SetupTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 118 of file equations_set_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_-
SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_-
STANDARD_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_-
EQUATIONS_SET_SETUP().

5.9.2.2 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_- SETUP_DEPENDENT_TYPE = 3

Dependent variables.

See also:

[EQUATIONS_SET_CONSTANTS::SetupTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 114 of file equations_set_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_-
SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_-
STANDARD_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_-
EQUATIONS_SET_SETUP().

5.9.2.3 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_- SETUP_EQUATIONS_TYPE = 9

Equations setup.

See also:

[EQUATIONS_SET_CONSTANTS::SetupTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 120 of file equations_set_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SETUP().

5.9.2.4 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FINAL_TYPE = 9

Final setup.

See also:

[EQUATIONS_SET_CONSTANTS::SetupTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 121 of file equations_set_constants.f90.

5.9.2.5 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FIXED_CONDITIONS_TYPE = 8

Fixed conditions.

See also:

[EQUATIONS_SET_CONSTANTS::SetupTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 119 of file equations_set_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SETUP().

5.9.2.6 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_GEOMETRY_TYPE = 2

Geometry setup.

See also:

[EQUATIONS_SET_CONSTANTS::SetupTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 113 of file equations_set_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SETUP().

5.9.2.7 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_INITIAL_TYPE = 1

Initial setup.

See also:

[EQUATIONS_SET_CONSTANTS::SetupTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 112 of file equations_set_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SETUP(), FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SUBTYPE_SET(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_SUBTYPE_SET(), LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SUBTYPE_SET().

5.9.2.8 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_MATERIALS_TYPE = 4

Materials setup.

See also:

[EQUATIONS_SET_CONSTANTS::SetupTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 115 of file equations_set_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SETUP().

5.9.2.9 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_SOURCE_MATERIALS_TYPE = 6

Source materials setup.

See also:

[EQUATIONS_SET_CONSTANTS::SetupTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 117 of file equations_set_constants.f90.

5.9.2.10 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_SOURCE_TYPE = 5

Source setup.

See also:

[EQUATIONS_SET_CONSTANTS::SetupTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 116 of file equations_set_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SETUP().

5.10 EQUATIONS_SET_CONSTANTS::SetupActionTypes

Setup action type parameters.

Variables

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_START_ACTION = 1](#)
Start setup action.
- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FINISH_ACTION = 2](#)
Finish setup action.

5.10.1 Detailed Description

Setup action type parameters.

See also:

[EQUATIONS_SET_CONSTANTS](#)

5.10.2 Variable Documentation

5.10.2.1 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FINISH_ACTION = 2

Finish setup action.

See also:

[EQUATIONS_SET_CONSTANTS::SetupActionTypes](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 129 of file equations_set_constants.f90.

Referenced by [FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SETUP\(\)](#), [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP\(\)](#), and [LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SETUP\(\)](#).

5.10.2.2 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_START_ACTION = 1

Start setup action.

See also:

[EQUATIONS_SET_CONSTANTS::SetupActionTypes](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 128 of file equations_set_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SETUP(), FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SUBTYPE_SET(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_SUBTYPE_SET(), LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SUBTYPE_SET().

5.11 EQUATIONS_SET_CONSTANTS::FixedConditions

Fixed conditons parameters.

Variables

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NOT_FIXED = 0](#)
The dof is not fixed.
- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FIXED_BOUNDARY_CONDITION = 1](#)
The dof is fixed as a boundary condition.
- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_MIXED_BOUNDARY_CONDITION = 2](#)
The dof is set as a mixed boundary condition.

5.11.1 Detailed Description

Fixed conditons parameters.

See also:

[EQUATIONS_SET_CONSTANTS](#)

5.11.2 Variable Documentation

5.11.2.1 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FIXED_BOUNDARY_CONDITION = 1

The dof is fixed as a boundary condition.

See also:

[EQUATIONS_SET_CONSTANTS::FixedConditions](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 137 of file equations_set_constants.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_PARAMETER_SET_UPDATE(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_FIXED_CONDITIONS_(), and SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE().

5.11.2.2 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_MIXED_BOUNDARY_CONDITION = 2

The dof is set as a mixed boundary condition.

See also:

[EQUATIONS_SET_CONSTANTS::FixedConditions](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 138 of file equations_set_constants.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_FIXED_CONDITIONS_(), and SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE().

5.11.2.3 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NOT_FIXED = 0

The dof is not fixed.

See also:

[EQUATIONS_SET_CONSTANTS::FixedConditions](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 136 of file equations_set_constants.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_FIXED_CONDITIONS_(), and SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE().

5.12 EQUATIONS_SET_CONSTANTS::LinearityTypes

The problem linearity type parameters.

Variables

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::NUMBER_OF_EQUATIONS_SET_LINEARITY_TYPES](#) = 3

The number of problem linearity types defined.

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LINEAR](#) = 1

The problem is linear.

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NONLINEAR](#) = 2

The problem is non-linear.

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NONLINEAR_BCS](#) = 3

The problem has non-linear boundary conditions.

5.12.1 Detailed Description

The problem linearity type parameters.

See also:

[EQUATIONS_SET_CONSTANTS](#)

5.12.2 Variable Documentation

5.12.2.1 INTEGER(INTG),parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LINEAR](#) = 1

The problem is linear.

See also:

[EQUATIONS_SET_CONSTANTS::LinearityTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 146 of file equations_set_constants.f90.

Referenced by [EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_CALCULATE\(\)](#), [EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_CREATE_VALUES_CACHE_INITIALISE\(\)](#), [EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_LINEAR_MATRICES_NUMBER_SET\(\)](#), [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP\(\)](#), and [LINEAR_ELASTICITY_ROUTINES::LINEAR_EQUATIONS_SET_SETUP\(\)](#).

**5.12.2.2 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
NONLINEAR = 2**

The problem is non-linear.

See also:

[EQUATIONS_SET_CONSTANTS::LinearityTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 147 of file equations_set_constants.f90.

Referenced by EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_RESIDUAL_-
VARIABLE_TYPE_SET(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_-
RHS_VARIABLE_TYPE_SET(), and FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_-
EQUATIONS_SET_SETUP().

**5.12.2.3 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
NONLINEAR_BCS = 3**

The problem has non-linear boundary conditions.

See also:

[EQUATIONS_SET_CONSTANTS::LinearityTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 148 of file equations_set_constants.f90.

**5.12.2.4 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::NUMBER_OF_-
EQUATIONS_SET_LINEARITY_TYPES = 3**

The number of problem linearity types defined.

See also:

[EQUATIONS_SET_CONSTANTS::LinearityTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 145 of file equations_set_constants.f90.

5.13 EQUATIONS_SET_CONSTANTS::JacobianCalculationTypes

The Jacobian types for nonlinear equations sets.

Variables

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-JACOBIAN_NOT_CALCULATED = 1](#)

The Jacobian values will not be calculated for the nonlinear equations set.

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-JACOBIAN_ANALYTIC_CALCULATED = 2](#)

The Jacobian values will be calculated analytically for the nonlinear equations set.

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-JACOBIAN_FD_CALCULATED = 3](#)

The Jacobian values will be calcualted using finite differences for the nonlinear equations set.

5.13.1 Detailed Description

The Jacobian types for nonlinear equations sets.

See also:

[EQUATIONS_SET_CONSTANTS](#)

5.13.2 Variable Documentation

- 5.13.2.1 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-JACOBIAN_ANALYTIC_CALCULATED = 2**

The Jacobian values will be calculated analytically for the nonlinear equations set.

See also:

[EQUATIONS_SET_CONSTANTS::JacobianCalculationTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 156 of file equations_set_constants.f90.

- 5.13.2.2 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-JACOBIAN_FD_CALCULATED = 3**

The Jacobian values will be calcualted using finite differences for the nonlinear equations set.

See also:

[EQUATIONS_SET_CONSTANTS::JacobianCalculationTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 157 of file equations_set_constants.f90.

**5.13.2.3 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
JACOBIAN_NOT_CALCULATED = 1**

The Jacobian values will not be calculated for the nonlinear equations set.

See also:

[EQUATIONS_SET_CONSTANTS::JacobianCalculationTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 155 of file `equations_set_constants.f90`.

5.14 EQUATIONS_SET_CONSTANTS::TimeDependenceTypes

The problem time dependence type parameters.

Variables

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::NUMBER_OF_EQUATIONS_SET_TIME_TYPES = 3](#)
The number of problem time dependence types defined.
- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_STATIC = 1](#)
The problem is static and has no time dependence.
- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_DYNAMIC = 2](#)
The problem is dynamic.
- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_QUASISTATIC = 3](#)
The problem is quasi-static.

5.14.1 Detailed Description

The problem time dependence type parameters.

See also:

[EQUATIONS_SET_CONSTANTS](#)

5.14.2 Variable Documentation

5.14.2.1 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_DYNAMIC = 2

The problem is dynamic.

See also:

[EQUATIONS_SET_CONSTANTS_TimeDependenceTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 167 of file equations_set_constants.f90.

5.14.2.2 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_QUASISTATIC = 3

The problem is quasi-static.

See also:

EQUATIONS_SET_CONSTANTS_TimeDependenceTypes,[EQUATIONS_SET_CONSTANTS](#)

Definition at line 168 of file equations_set_constants.f90.

5.14.2.3 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_- STATIC = 1

The problem is static and has no time dependence.

See also:

EQUATIONS_SET_CONSTANTS_TimeDependenceTypes,[EQUATIONS_SET_CONSTANTS](#)

Definition at line 166 of file equations_set_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_-
SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_-
STANDARD_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_-
EQUATIONS_SET_SETUP().

5.14.2.4 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::NUMBER_OF_- EQUATIONS_SET_TIME_TYPES = 3

The number of problem time dependence types defined.

See also:

EQUATIONS_SET_CONSTANTS_TimeDependenceTypes,[EQUATIONS_SET_CONSTANTS](#)

Definition at line 165 of file equations_set_constants.f90.

5.15 EQUATIONS_SET_CONSTANTS::SolutionMethods

The solution method parameters.

Variables

- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::NUMBER_OF_EQUATIONS_SET_SOLUTION_METHOD = 6`
The number of solution methods defined.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FEM_SOLUTION_METHOD = 1`
Finite Element Method solution method.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_BEM_SOLUTION_METHOD = 2`
Boundary Element Method solution method.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FD_SOLUTION_METHOD = 3`
Finite Difference solution method.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FV_SOLUTION_METHOD = 4`
Finite Volume solution method.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_GFEM_SOLUTION_METHOD = 5`
Grid-based Finite Element Method solution method.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_GFV_SOLUTION_METHOD = 6`
Grid-based Finite Volume solution method.

5.15.1 Detailed Description

The solution method parameters.

See also:

[EQUATIONS_SET_CONSTANTS](#)

5.15.2 Variable Documentation

5.15.2.1 INTEGER(INTG),parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_BEM_SOLUTION_METHOD = 2`

Boundary Element Method solution method.

See also:

[EQUATIONS_SET_CONSTANTS::SolutionMethods](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 177 of file equations_set_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_-
SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_-
STANDARD_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_-
EQUATIONS_SET_SETUP().

5.15.2.2 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_- FD_SOLUTION_METHOD = 3

Finite Difference solution method.

See also:

[EQUATIONS_SET_CONSTANTS::SolutionMethods](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 178 of file equations_set_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_-
SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_-
STANDARD_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_-
EQUATIONS_SET_SETUP().

5.15.2.3 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_- FEM_SOLUTION_METHOD = 1

Finite Element Method solution method.

See also:

[EQUATIONS_SET_CONSTANTS::SolutionMethods](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 176 of file equations_set_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_-
SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_-
STANDARD_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_-
EQUATIONS_SET_SETUP().

5.15.2.4 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_- FV_SOLUTION_METHOD = 4

Finite Volume solution method.

See also:

[EQUATIONS_SET_CONSTANTS::SolutionMethods](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 179 of file equations_set_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_-
SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_-
STANDARD_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_-
EQUATIONS_SET_SETUP().

**5.15.2.5 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
GFEM_SOLUTION_METHOD = 5**

Grid-based Finite Element Method solution method.

See also:

[EQUATIONS_SET_CONSTANTS::SolutionMethods](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 180 of file equations_set_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_-
SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_-
STANDARD_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_-
EQUATIONS_SET_SETUP().

**5.15.2.6 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
GFV_SOLUTION_METHOD = 6**

Grid-based Finite Volume solution method.

See also:

[EQUATIONS_SET_CONSTANTS::SolutionMethods](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 181 of file equations_set_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_-
SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_-
STANDARD_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_-
EQUATIONS_SET_SETUP().

**5.15.2.7 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::NUMBER_OF_-
EQUATIONS_SET_SOLUTION_METHODS = 6**

The number of solution methods defined.

See also:

[EQUATIONS_SET_CONSTANTS::SolutionMethods](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 175 of file equations_set_constants.f90.

5.16 EQUATIONS_SET_CONSTANTS::OutputTypes

Equations output types.

Variables

- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NO_OUTPUT = 0**

No output.

- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_TIMING_OUTPUT = 1**

Timing information output.

- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_MATRIX_OUTPUT = 2**

All below and equation matrices output.

- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_ELEMENT_MATRIX_OUTPUT = 3**

All below and Element matrices output.

5.16.1 Detailed Description

Equations output types.

See also:

[EQUATIONS_SET_CONSTANTS](#)

5.16.2 Variable Documentation

5.16.2.1 INTEGER(INTG),parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_ELEMENT_MATRIX_OUTPUT = 3**

All below and Element matrices output.

See also:

[EQUATIONS_SET_CONSTANTS::OutputTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 191 of file equations_set_constants.f90.

5.16.2.2 INTEGER(INTG),parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_MATRIX_OUTPUT = 2**

All below and equation matrices output.

See also:

[EQUATIONS_SET_CONSTANTS::OutputTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 190 of file equations_set_constants.f90.

**5.16.2.3 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
NO_OUTPUT = 0**

No output.

See also:

[EQUATIONS_SET_CONSTANTS::OutputTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 188 of file equations_set_constants.f90.

**5.16.2.4 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
TIMING_OUTPUT = 1**

Timing information output.

See also:

[EQUATIONS_SET_CONSTANTS::OutputTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 189 of file equations_set_constants.f90.

5.17 EQUATIONS_SET_CONSTANTS::SparsityTypes

Equations matrices sparsity types.

Variables

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SPARSE_MATRICES = 1](#)

Use sparse matrices for the equations set.

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FULL_MATRICES = 2](#)

Use fully populated matrices for the equations set.

5.17.1 Detailed Description

Equations matrices sparsity types.

See also:

[EQUATIONS_SET_CONSTANTS](#)

5.17.2 Variable Documentation

5.17.2.1 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FULL_MATRICES = 2

Use fully populated matrices for the equations set.

See also:

[EQUATIONS_SET_CONSTANTS::SparsityTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 199 of file equations_set_constants.f90.

5.17.2.2 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SPARSE_MATRICES = 1

Use sparse matrices for the equations set.

See also:

[EQUATIONS_SET_CONSTANTS::SparsityTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 198 of file equations_set_constants.f90.

5.18 FIELD_ROUTINES::DependentTypes

Depedent field parameter types.

Functions

- subroutine [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_CALCULATE](#) (FIELD, EQUATION_NUMBER, GEOMETRIC_PARAMETERS, VARIABLE_NUMBER, COMPONENT_NUMBER, NODE_NUMBER, DERIV_NUMBER, VALUE, ERR, ERROR,*)

Calculates the element stiffness matrices and RHS for a Laplace equation finite element equations set.

- subroutine [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_INTE](#) (FIELD, NODE_NUMBER, COMPONENT_NUMBER, VARIABLE_NUMBER, NUMBER_OF_DIMENSIONS,&INTERPOLATED_POINT, ERR, ERROR,*)

Gets the intepolated points for the particular node. Assume the node allocation for an element is fixed.

- subroutine [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_FIXED_CONDITIONS_](#) (EQUATIONS_SET, SET_TYPE, DERIVATIVE_NUMBER, NODE_NUMBER, COMPONENT_NUMBER,&VARIABLE_NUMBER, CONDITION, VALUE, ERR, ERROR,*)

Sets a fixed condition for the equation set on the specified node. This is more generic method, move to other place.

- subroutine [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_PARAMETER_SET_UPDATE](#) (EQUATIONS_SET, GEOMETRIC_PARAMETERS, NODE_TYPE, ERR, ERROR,*)

Calculates the element stiffness matrices and RHS for a Laplace equation finite element equations set.

- subroutine [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATIONFINITE_ELEMENT_CALCULATE](#) (EQUATIONS_SET, ELEMENT_NUMBER, ERR, ERROR,*)

Calculates the element stiffness matrices and RHS for a Laplace equation finite element equations set.

- subroutine [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_SETUP](#) (EQUATIONS_SET, SETUP_TYPE, ACTION_TYPE, ERR, ERROR,*)

Sets up the Laplace equation type of a classical field equations set class.

- subroutine [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_SUBTYPE_SET](#) (EQUATIONS_SET, EQUATIONS_SET_SUBTYPE, ERR, ERROR,*)

Sets/changes the equation subtype for a Laplace equation type of a classical field equations set class.

- subroutine [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP](#) (EQUATIONS_SET, SETUP_TYPE, ACTION_TYPE, ERR, ERROR,*)

Sets up the standard Laplace equation.

- subroutine [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SETUP](#) (PROBLEM, SETUP_TYPE, ACTION_TYPE, ERR, ERROR,*)

Sets up the Laplace solution.

- subroutine [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SUBTYPE_SET](#) (PROBLEM, PROBLEM_SUBTYPE, ERR, ERROR,*)

Sets/changes the problem subtype for a Laplace equation type .

- subroutine [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP](#)(PROBLEM, SETUP_TYPE, ACTION_TYPE, ERR, ERROR,*)

Sets up the standard Laplace equations solution.

Variables

- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_INDEPENDENT_TYPE](#) = 1
Independent field type.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_DEPENDENT_TYPE](#) = 2
Dependent field type.
- INTEGER(INTG), parameter [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_TWO_DIM_1](#) = 1
$$u=x^{**2}+2*x*y-y^{**2}$$
- INTEGER(INTG), parameter [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_TWO_DIM_2](#) = 2
$$u=\cos(x)\cosh(y)$$
- INTEGER(INTG), parameter [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_THREE_DIM_1](#) = 3
$$u=x^{**2}-2*y^{**2}+z^{**2}$$
- INTEGER(INTG), parameter [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_THREE_DIM_2](#) = 4
$$u=\cos(x)*\cosh(y)*z$$

5.18.1 Detailed Description

Depedent field parameter types.

the analytic representive of laplace equation

See also:

[FIELD_ROUTINES](#)
[LAPLACE_EQUATIONS_ROUTINES](#)

5.18.2 Function Documentation

5.18.2.1 subroutine LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_CALCULATE (TYPE(FIELD_TYPE),pointer *FIELD*,
 INTEGER(INTG),intent(in) *EQUATION_NUMBER*, REAL(DP),dimension(:),intent(in)
GEOMETRIC_PARAMETERS, INTEGER(INTG),intent(in) *VARIABLE_NUMBER*,
 INTEGER(INTG),intent(in) *COMPONENT_NUMBER*, INTEGER(INTG),intent(in)
NODE_NUMBER, INTEGER(INTG),intent(in) *DERIV_NUMBER*,
 REAL(DP),intent(out) *VALUE*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Calculates the element stiffness matrices and RHS for a Laplace equation finite element equations set.

Todo

for regular mesh only

Parameters:

ERR The error code

ERROR The error string

Definition at line 106 of file Laplace_equations_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_FINALISE(), FIELD_-ROUTINES::FIELD_NORMAL_VARIABLE_TYPE, FIELD_ROUTINES::FIELD_STANDARD_-VARIABLE_TYPE, LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_THREE_DIM_1, LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_THREE_DIM_2, LAPLACE_-EQUATIONS_ROUTINES::LAPLACE_EQUATION_TWO_DIM_1, and LAPLACE_EQUATIONS_-ROUTINES::LAPLACE_EQUATION_TWO_DIM_2.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_-PARAMETER_SET_UPDATE().

Here is the call graph for this function:

Here is the caller graph for this function:

5.18.2.2 subroutine LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_PARAMETER_SET_UPDATE (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, REAL(DP),dimension(:),intent(in) *GEOMETRIC_PARAMETERS*,
 INTEGER(INTG),intent(in) *NODE_TYPE*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Calculates the element stiffness matrices and RHS for a Laplace equation finite element equations set.

Parameters:

ERR The error code

ERROR The error string

Definition at line 439 of file Laplace_equations_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-FIXED_BOUNDARY_CONDITION, BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(),

`FIELD_ROUTINES::FIELD_ANALYTIC_SET_TYPE, FIELD_ROUTINES::FIELD_BOUNDARY_CONDITIONS_SET_TYPE, and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_CALCULATE()`.

Referenced by `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP()`.

Here is the call graph for this function:

Here is the caller graph for this function:

5.18.2.3 subroutine LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_SETUP (TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET, INTEGER(INTG),intent(in) SETUP_TYPE, INTEGER(INTG),intent(in) ACTION_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Sets up the Laplace equation type of a classical field equations set class.

Parameters:

EQUATIONS_SET A pointer to the equations set to setup a Laplace equation on.

SETUP_TYPE The setup type

ACTION_TYPE The action type

ERR The error code

ERROR The error string

Definition at line 626 of file Laplace_equations_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_GENERALISED_LAPLACE_SUBTYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_STANDARD_LAPLACE_SUBTYPE`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP()`.

Referenced by `CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SET_SETUP()`.

Here is the call graph for this function:

Here is the caller graph for this function:

5.18.2.4 subroutine LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP (TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET, INTEGER(INTG),intent(in) SETUP_TYPE, INTEGER(INTG),intent(in) ACTION_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Sets up the standard Laplace equation.

Parameters:

EQUATIONS_SET A pointer to the equations set to setup

SETUP_TYPE The setup type to perform

ACTION_TYPE The action type to perform

ERR The error code

ERROR The error string

Definition at line 709 of file Laplace_equations_routines.f90.

```
References BASE_ROUTINES::ENTERS(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_
MAPPING_CREATE_FINISH(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_
MAPPING_CREATE_START(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_
LINEAR_MATRICES_NUMBER_SET(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_
MAPPING_LINEAR_MATRICES_VARIABLE_TYPES_SET(), EQUATIONS_MAPPING_
ROUTINES::EQUATIONS_MAPPING_RHS_VARIABLE_TYPE_SET(), EQUATIONS_
SET_CONSTANTS::EQUATIONS_SET_BEM SOLUTION_METHOD, EQUATIONS_
SET_CONSTANTS::EQUATIONS_SET_FD SOLUTION_METHOD, EQUATIONS_SET_
CONSTANTS::EQUATIONS_SET_FEM SOLUTION_METHOD, EQUATIONS_SET_
CONSTANTS::EQUATIONS_SET_FV SOLUTION_METHOD, EQUATIONS_SET_
CONSTANTS::EQUATIONS_SET_GFEM SOLUTION_METHOD, EQUATIONS_SET_
CONSTANTS::EQUATIONS_SET_GFV SOLUTION_METHOD, EQUATIONS_SET_
CONSTANTS::EQUATIONS_SET_LINEAR, EQUATIONS_SET_CONSTANTS::EQUATIONS_
SET_SETUP_ANALYTIC_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_
SETUP_DEPENDENT_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_
SETUP_EQUATIONS_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_
SETUP_FINISH_ACTION, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_
FIXED_CONDITIONS_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_
SETUP_GEOMETRY_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_
SETUP_INITIAL_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_
MATERIALS_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_
SOURCE_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_START_
ACTION, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_STANDARD_LAPLACE_
SUBTYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_STATIC, BASE_
ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), FIELD_ROUTINES::FIELD_ANALYTIC_
SET_TYPE, FIELD_ROUTINES::FIELD_BOUNDARY_CONDITIONS_SET_TYPE, FIELD_
ROUTINES::FIELD_CREATE_FINISH(), FIELD_ROUTINES::FIELD_DEPENDENT_TYPE,
FIELD_ROUTINES::FIELD_GENERAL_TYPE, FIELD_ROUTINES::FIELD_NODE_BASED_
INTERPOLATION, FIELD_ROUTINES::FIELD_NORMAL_VARIABLE_TYPE, FIELD_
ROUTINES::FIELD_STANDARD_VARIABLE_TYPE, FIELD_ROUTINES::FIELD_VALUES_SET_
TYPE, LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_PARAMETER_
SET_UPDATE(), MATRIX_VECTOR::MATRIX_BLOCK_STORAGE_TYPE, and MATRIX_
VECTOR::MATRIX_COMPRESSED_ROW_STORAGE_TYPE.
```

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_
SETUP(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_
SUBTYPE_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

**5.18.2.5 subroutine LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION-
EQUATIONS_SET_SUBTYPE_SET (TYPE(EQUATIONS_SET_TYPE),pointer
EQUATIONS_SET, INTEGER(INTG),intent(in) EQUATIONS_SET_SUBTYPE,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)
[private]**

Sets/changes the equation subtype for a Laplace equation type of a classical field equations set class.

Parameters:

EQUATIONS_SET A pointer to the equations set to set the equation subtype for

EQUATIONS_SET_SUBTYPE The equation subtype to set

ERR The error code

ERROR The error string

Definition at line 666 of file Laplace_equations_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_CLASSICAL_FIELD_CLASS`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_GENERALISED LAPLACE_SUBTYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET LAPLACE_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_INITIAL_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_START_ACTION`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_STANDARD LAPLACE_SUBTYPE`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP()`.

Referenced by `CLASSICAL_FIELD_ROUTINES::CL()`.

Here is the call graph for this function:

Here is the caller graph for this function:

5.18.2.6 subroutine LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATIONFINITE_ELEMENT_CALCULATE (TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET, INTEGER(INTG),intent(in) ELEMENT_NUMBER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Calculates the element stiffness matrices and RHS for a Laplace equation finite element equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to perform the finite element calculations on

ELEMENT_NUMBER The element number to calculate

ERR The error code

ERROR The error string

Definition at line 512 of file Laplace_equations_routines.f90.

References `KINDS::_DP`, `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_GENERALISED LAPLACE_SUBTYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_STANDARD LAPLACE_SUBTYPE`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_ROUTINES::FIELD_INTERPOLATE_GAUSS()`, `FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_CALCULATE()`, `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET()`, `FIELD_ROUTINES::FIELD_VALUES_SET_TYPE`, and `KINDS::PTR`.

Referenced by `CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELDFINITE_ELEMENT_CALCULATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**5.18.2.7 subroutine LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_FIXED_-
CONDITIONS_(TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET,
INTEGER(INTG),intent(in) SET_TYPE, INTEGER(INTG),intent(in)
DERIVATIVE_NUMBER, INTEGER(INTG),intent(in) NODE_NUMBER,
INTEGER(INTG),intent(in) COMPONENT_NUMBER, &,intent(in)
VARIABLE_NUMBER, INTEGER(INTG),intent(in) condition, REAL(DP),intent(in)
VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)
ERROR, *) [private]**

Sets a fixed condition for the equation set on the specified node. This is more generic method, move to other place.

Parameters:

EQUATIONS_SET A pointer to the equations set to set the fixed condition for
SET_TYPE The field parameter set type
DERIVATIVE_NUMBER The derivative to set the fixed condition at
NODE_NUMBER The node_number to set the fixed condition at
COMPONENT_NUMBER The component number to set the fixed condition at
VALUE The value of the fixed condition to set
ERR The error code
ERROR The error string

Definition at line 352 of file Laplace_equations_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_-
SET_FIXED_BOUNDARY_CONDITION, EQUATIONS_SET_CONSTANTS::EQUATIONS_-
SET_MIXED_BOUNDARY_CONDITION, EQUATIONS_SET_CONSTANTS::EQUATIONS_-
SET_NOT_FIXED, BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and NODE_-
ROUTINES::NODE_CHECK_EXISTS().

Here is the call graph for this function:

**5.18.2.8 subroutine LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_INTE
(TYPE(FIELD_TYPE),pointer FIELD, INTEGER(INTG),intent(in) NODE_NUMBER,
INTEGER(INTG),intent(in) COMPONENT_NUMBER, INTEGER(INTG),intent(in)
VARIABLE_NUMBER, INTEGER(INTG),intent(in) NUMBER_OF_DIMENSIONS,
&,pointer INTERPOLATED_POINT, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Gets the intepolated points for the particular node. Assume the node allocation for an element is fixed.

Todo

this is more generic, move to other place.

Parameters:

ERR The error code
ERROR The error string

Definition at line 263 of file Laplace_equations_routines.f90.

References KINDS::DP, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(),
 BASE_ROUTINES::EXITS(), FIELD_ROUTINES::FIELD_INTERPOLATE_XI(), FIELD_-
 ROUTINES::FIELD_INTERPOLATED_POINT_INITIALISE(), FIELD_-
 INTERPOLATION_PARAMETERS_ELEMENT_GET(), FIELD_-
 INTERPOLATION_PARAMETERS_FINALISE(), and FIELD_-
 INTERPOLATION_PARAMETERS_INITIALISE().

Here is the call graph for this function:

**5.18.2.9 subroutine LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-
 PROBLEM_SETUP (TYPE(PROBLEM_TYPE),pointer *PROBLEM*,
 INTEGER(INTG),intent(in) *SETUP_TYPE*, INTEGER(INTG),intent(in) *ACTION_-
 TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out)
ERROR, *)**

Sets up the Laplace solution.

Parameters:

PROBLEM A pointer to the solutions set to setup a Laplace equation on.
SETUP_TYPE The setup type
ACTION_TYPE The action type
ERR The error code
ERROR The error string

Definition at line 986 of file Laplace_equations_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(),
 BASE_ROUTINES::EXITS(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_-
 STANDARD_SETUP(), PROBLEM_CONSTANTS::PROBLEM_GENERALISED_LAPLACE_-
 SUBTYPE, and PROBLEM_CONSTANTS::PROBLEM_STANDARD_LAPLACE_SUBTYPE.

Referenced by CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_SETUP().

Here is the call graph for this function:

Here is the caller graph for this function:

**5.18.2.10 subroutine LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-
 PROBLEM_STANDARD_SETUP (TYPE(PROBLEM_TYPE),pointer *PROBLEM*,
 INTEGER(INTG),intent(in) *SETUP_TYPE*, INTEGER(INTG),intent(in) *ACTION_-
 TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out)
ERROR, *) [private]**

Sets up the standard Laplace equations solution.

Parameters:

PROBLEM A pointer to the problem to setup
SETUP_TYPE The setup type to perform
ACTION_TYPE The action type to perform
ERR The error code
ERROR The error string

Definition at line 1069 of file Laplace_equations_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_CLASSICAL_FIELD_CLASS`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LAPLACE_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_STANDARD_LAPLACE_SUBTYPE`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_ROUTINES::FIELD_STANDARD_VARIABLE_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_SETUP_CONTROL_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_SETUP_DO_ACTION`, `PROBLEM_CONSTANTS::PROBLEM_SETUP_FINISH_ACTION`, `PROBLEM_CONSTANTS::PROBLEM_SETUP_INITIAL_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_SETUP_SOLUTION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_SETUP_SOLVER_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_SETUP_START_ACTION`, `PROBLEM_CONSTANTS::PROBLEM_STANDARD_LAPLACE_SUBTYPE`, `KINDS::PTR`, `SOLVER_ROUTINES::SOLVER_CREATE_FINISH()`, `SOLVER_ROUTINES::SOLVER_CREATE_START()`, `SOLVER_ROUTINES::SOLVER_LIBRARY_SET()`, `SOLVER_ROUTINES::SOLVER_LINEAR_TYPE`, `SOLVER_ROUTINES::SOLVER_PETSC_LIBRARY`, `SOLVER_ROUTINES::SOLVER_SPARSE_MATRICES`, and `SOLVER_ROUTINES::SOLVER_SPARSITY_TYPE_SET()`.

Referenced by `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SETUP()`, and `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SUBTYPE_SET()`.

Here is the call graph for this function:

Here is the caller graph for this function:

5.18.2.11 subroutine LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SUBTYPE_SET (TYPE(PROBLEM_TYPE),pointer **PROBLEM**, INTEGER(INTG),intent(in) **PROBLEM_SUBTYPE**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING_STRING),intent(out) **ERROR**, *)

Sets/changes the problem subtype for a Laplace equation type .

Parameters:

PROBLEM A pointer to the problem to set the problem subtype for

PROBLEM_SUBTYPE The problem subtype to set

ERR The error code

ERROR The error string

Definition at line 1026 of file Laplace_equations_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP()`, `PROBLEM_CONSTANTS::PROBLEM_CLASSICAL_FIELD_CLASS`, `PROBLEM_CONSTANTS::PROBLEM_GENERALISED_LAPLACE_SUBTYPE`, `PROBLEM_CONSTANTS::PROBLEM_LAPLACE_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_SETUP_INITIAL_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_SETUP_START_ACTION`, and `PROBLEM_CONSTANTS::PROBLEM_STANDARD_LAPLACE_SUBTYPE`.

Referenced by `CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_SET()`.

Here is the call graph for this function:

Here is the caller graph for this function:

5.18.3 Variable Documentation

5.18.3.1 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_DEPENDENT_TYPE = 2

Dependent field type.

See also:

[FIELD_ROUTINES::DependentTypes](#),[FIELD_ROUTINES](#)

Definition at line 74 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET_PTR(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SETUP().

5.18.3.2 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_INDEPENDENT_TYPE = 1

Independent field type.

See also:

[FIELD_ROUTINES::DependentTypes](#),[FIELD_ROUTINES](#)

Definition at line 73 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FIELD_CREATE_FINISH(), FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET_PTR(), and FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

5.18.3.3 INTEGER(INTG),parameter LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_THREE_DIM_1 = 3

$u=x^{**2}-2*y^{**2}+z^{**2}$

Definition at line 81 of file Laplace_equations_routines.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_CALCULATE().

5.18.3.4 INTEGER(INTG),parameter LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_THREE_DIM_2 = 4

$u=\cos(x)*\cosh(y)*z$

Definition at line 82 of file Laplace_equations_routines.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_CALCULATE().

5.18.3.5 INTEGER(INTG),parameter LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_TWO_DIM_1 = 1

$u=x^{**2}+2*x*y-y^{**2}$

Definition at line 79 of file Laplace_equations_routines.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_-CALCULATE().

5.18.3.6 INTEGER(INTG),parameter LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_TWO_DIM_2 = 2

$u = \cos(x) \cosh(y)$

Definition at line 80 of file Laplace_equations_routines.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_-CALCULATE().

5.19 FIELD_ROUTINES::DimensionTypes

Field dimension parameter types.

Variables

- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_SCALAR_DIMENSION_TYPE](#) = 1
Scalar field.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_VECTOR_DIMENSION_TYPE](#) = 2
Vector field.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_TENSOR_DIMENSION_TYPE](#) = 2
Tensor field.

5.19.1 Detailed Description

Field dimension parameter types.

See also:

[FIELD_ROUTINES](#)

5.19.2 Variable Documentation

5.19.2.1 INTEGER(INTG),parameter [FIELD_ROUTINES::FIELD_SCALAR_DIMENSION_TYPE](#) = 1

Scalar field.

See also:

[FIELD_ROUTINES::DimensionTypes](#),[FIELD_ROUTINES](#)

Definition at line 81 of file `field_routines.f90`.

Referenced by `FIELD_ROUTINES::FIELD_DIMENSION_SET_PTR()`, and `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`.

5.19.2.2 INTEGER(INTG),parameter [FIELD_ROUTINES::FIELD_TENSOR_DIMENSION_TYPE](#) = 2

Tensor field.

See also:

[FIELD_ROUTINES::DimensionTypes](#),[FIELD_ROUTINES](#)

Definition at line 83 of file `field_routines.f90`.

**5.19.2.3 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_VECTOR_DIMENSION_-
TYPE = 2**

Vector field.

See also:

[FIELD_ROUTINES::DimensionTypes](#),[FIELD_ROUTINES](#)

Definition at line 82 of file `field_routines.f90`.

Referenced by `FIELD_ROUTINES::FIELD_CREATE_FINISH()`, `FIELD_ROUTINES::FIELD_-
DIMENSION_SET_PTR()`, and `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_-
LINE_GET()`.

5.20 FIELD_ROUTINES::FieldTypes

Field type parameters.

Variables

- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_GEOMETRIC_TYPE = 1](#)
Geometric field.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_FIBRE_TYPE = 2](#)
Fibre field.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_GENERAL_TYPE = 3](#)
General field.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_MATERIAL_TYPE = 4](#)
Material field.

5.20.1 Detailed Description

Field type parameters.

See also:

[FIELD_ROUTINES](#)

5.20.2 Variable Documentation

5.20.2.1 INTEGER(INTG),parameter [FIELD_ROUTINES::FIELD_FIBRE_TYPE = 2](#)

Fibre field.

See also:

[FIELD_ROUTINES::FieldTypes](#),[FIELD_ROUTINES](#)

Definition at line 91 of file field_routines.f90.

Referenced by [FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_INITIALISE\(\)](#), [FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_CALCULATE\(\)](#), [FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET\(\)](#), [FIELD_IO_ROUTINES::FIELD_IO_FIELD_INFO\(\)](#), and [FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET\(\)](#).

5.20.2.2 INTEGER(INTG),parameter [FIELD_ROUTINES::FIELD_GENERAL_TYPE = 3](#)

General field.

See also:

[FIELD_ROUTINES::FieldTypes](#),[FIELD_ROUTINES](#)

Definition at line 92 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET(), FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITYFINITE_ELEMENT_RESIDUAL_EVALUATE(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SETUP().

5.20.2.3 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_GEOMETRIC_TYPE = 1

Geometric field.

See also:

[FIELD_ROUTINES::FieldTypes](#),[FIELD_ROUTINES](#)

Definition at line 90 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FIELD_CREATE_FINISH(), FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_CALCULATE(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), FIELD_IO_ROUTINES::FIELD_IO_FIELD_INFO(), FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET(), and FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SETUP().

5.20.2.4 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_MATERIAL_TYPE = 4

Material field.

See also:

[FIELD_ROUTINES::FieldTypes](#),[FIELD_ROUTINES](#)

Definition at line 93 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), and FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET().

5.21 FIELD_ROUTINES::InterpolationTypes

Field interpolation parameters.

Variables

- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_CONSTANT_INTERPOLATION = 1**
Constant interpolation. One parameter for the field.
- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_ELEMENT_BASED_INTERPOLATION = 2**
Element based interpolation. Parameters are different in each element.
- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_NODE_BASED_INTERPOLATION = 3**
Node based interpolation. Parameters are nodal based and a basis function is used.
- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_GRID_POINT_BASED_INTERPOLATION = 4**
Grid point based interpolation. Parameters are different at each grid point.
- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_GAUSS_POINT_BASED_INTERPOLATION = 5**
Gauss point based interpolation. Parameters are different at each Gauss point.

5.21.1 Detailed Description

Field interpolation parameters.

See also:

[FIELD_ROUTINES](#)

5.21.2 Variable Documentation

5.21.2.1 INTEGER(INTG),parameter **FIELD_ROUTINES::FIELD_CONSTANT_INTERPOLATION = 1**

Constant interpolation. One parameter for the field.

See also:

[FIELD_ROUTINES::InterpolationTypes](#),[FIELD_ROUTINES](#)

Definition at line 100 of file `field_routines.f90`.

Referenced by `FIELD_ROUTINES::FI()`, `FIELD_ROUTINES::FIELD_COMPONENT_MESH_COM()`, `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET()`, `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`, and `FIELD_ROUTINES::FIELD_VARIABLE_COMPONENT_INITIALISE()`.

5.21.2.2 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_ELEMENT_BASED_- INTERPOLATION = 2

Element based interpolation. Parameters are different in each element.

See also:

[FIELD_ROUTINES::InterpolationTypes](#),[FIELD_ROUTINES](#)

Definition at line 101 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FI(), FIELD_ROUTINES::FIELD_COMPONENT_MESH_COM(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), FIELD_ROUTINES::FIELD_VARIABLE_COMPONENT_INITIALISE(), and FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITYFINITE_ELEMENT_RESIDUAL_EVALUATE().

5.21.2.3 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_GAUSS_POINT_BASED_- INTERPOLATION = 5

Gauss point based interpolation. Parameters are different at each Gauss point.

See also:

[FIELD_ROUTINES::InterpolationTypes](#),[FIELD_ROUTINES](#)

Definition at line 104 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FI(), FIELD_ROUTINES::FIELD_COMPONENT_MESH_COM(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), and FIELD_ROUTINES::FIELD_VARIABLE_COMPONENT_INITIALISE().

5.21.2.4 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_GRID_POINT_BASED_- INTERPOLATION = 4

Grid point based interpolation. Parameters are different at each grid point.

See also:

[FIELD_ROUTINES::InterpolationTypes](#),[FIELD_ROUTINES](#)

Definition at line 103 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FI(), FIELD_ROUTINES::FIELD_COMPONENT_MESH_COM(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), and FIELD_ROUTINES::FIELD_VARIABLE_COMPONENT_INITIALISE().

5.21.2.5 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_NODE_BASED_- INTERPOLATION = 3

Node based interpolation. Parameters are nodal based and a basis function is used.

See also:

[FIELD_ROUTINES::InterpolationTypes](#), [FIELD_ROUTINES](#)

Definition at line 102 of file field_routines.f90.

Referenced by [FIELD_ROUTINES::FI\(\)](#), [FIELD_ROUTINES::FIELD_COMPONENT_MESH_COM\(\)](#), [FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_INITIALISE\(\)](#), [FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET\(\)](#), [FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET\(\)](#), [FIELD_ROUTINES::FIELD_VARIABLE_COMPONENT_INITIALISE\(\)](#), [FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SETUP\(\)](#), [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP\(\)](#), and [LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SETUP\(\)](#).

5.22 FIELD_ROUTINES::VariableTypes

Field variable type parameters.

Variables

- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_NUMBER_OF_VARIABLE_TYPES = 4`
Number of different field variable types possible.
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_STANDARD_VARIABLE_TYPE = 1`
Standard variable type i.e., u .
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_NORMAL_VARIABLE_TYPE = 2`
Normal derivative variable type i.e., du/dn .
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_TIME_DERIV1_VARIABLE_TYPE = 3`
First time derivative variable type i.e., du/dt .
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_TIME_DERIV2_VARIABLE_TYPE = 4`
Second type derivative variable type i.e., d^2u/dt^2 .

5.22.1 Detailed Description

Field variable type parameters.

See also:

[FIELD_ROUTINES](#)

Todo

sort out variable access routines so that you are always accessing by variable type rather than variable number.

5.22.2 Variable Documentation

5.22.2.1 INTEGER(INTG),parameter `FIELD_ROUTINES::FIELD_NORMAL_VARIABLE_TYPE = 2`

Normal derivative variable type i.e., du/dn .

See also:

[FIELD_ROUTINES::VariableTypes](#),[FIELD_ROUTINES](#)

Definition at line 114 of file field_routines.f90.

Referenced by EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_CREATE_VALUES_CACHE_INITIALISE(), FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET(), FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SETUP(), LAPLACE_EQUIVATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_CALCULATE(), LAPLACE_EQUIVATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SETUP().

5.22.2.2 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_NUMBER_OF_VARIABLE_TYPES = 4

Number of different field variable types possible.

See also:

[FIELD_ROUTINES::VariableTypes](#), [FIELD_ROUTINES](#)

Definition at line 112 of file field_routines.f90.

Referenced by EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_CALCULATE(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_LINEAR_MATRICES_NUMBER_SET(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_LINEAR_MATRICES_VARIABLE_TYPES_SET(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_RESIDUAL_VARIABLE_TYPE_SET(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_RHS_VARIABLE_TYPE_SET(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_SOURCE_VARIABLE_TYPE_SET(), FIELD_ROUTINES::FIELD_CREATE_FINISH(), and FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_INITIALISE().

5.22.2.3 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_STANDARD_VARIABLE_TYPE = 1

Standard variable type i.e., u.

See also:

[FIELD_ROUTINES::VariableTypes](#), [FIELD_ROUTINES](#)

Definition at line 113 of file field_routines.f90.

Referenced by EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_CREATE_VALUES_CACHE_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET(), FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SETUP(), LAPLACE_EQUIVATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_CALCULATE(), LAPLACE_EQUIVATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), LAPLACE_EQUIVATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(), LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SETUP().

5.22.2.4 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_TIME_DERIV1_VARIABLE_TYPE = 3

First time derivative variable type i.e., du/dt.

See also:

[FIELD_ROUTINES::VariableTypes](#),[FIELD_ROUTINES](#)

Definition at line 115 of file field_routines.f90.

Referenced by [FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET\(\)](#).

5.22.2.5 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_TIME_DERIV2_- VARIABLE_TYPE = 4

Second type derivative variable type i.e., d^2u/dt^2 .

See also:

[FIELD_ROUTINES::VariableTypes](#),[FIELD_ROUTINES](#)

Definition at line 116 of file field_routines.f90.

Referenced by [FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET\(\)](#).

5.23 FIELD_ROUTINES::DofTypes

Field dof type parameters.

Variables

- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_CONSTANT_DOF_TYPE = 1](#)
The dof is from a field variable component with constant interpolation.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_ELEMENT_DOF_TYPE = 2](#)
The dof is from a field variable component with element based interpolation.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_NODE_DOF_TYPE = 3](#)
The dof is from a field variable component with node based interpolation.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_POINT_DOF_TYPE = 4](#)
The dof is from a field variable component with point based interpolation.

5.23.1 Detailed Description

Field dof type parameters.

See also:

[FIELD_ROUTINES](#)

5.23.2 Variable Documentation

5.23.2.1 INTEGER(INTG),parameter [FIELD_ROUTINES::FIELD_CONSTANT_DOF_TYPE = 1](#)

The dof is from a field variable component with constant interpolation.

See also:

[FIELD_ROUTINES::DofTypes](#),[FIELD_ROUTINES](#)

Definition at line 123 of file `field_routines.f90`.

Referenced by `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`.

5.23.2.2 INTEGER(INTG),parameter [FIELD_ROUTINES::FIELD_ELEMENT_DOF_TYPE = 2](#)

The dof is from a field variable component with element based interpolation.

See also:

[FIELD_ROUTINES::DofTypes](#),[FIELD_ROUTINES](#)

Definition at line 124 of file `field_routines.f90`.

Referenced by `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`.

5.23.2.3 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_NODE_DOF_TYPE = 3

The dof is from a field variable component with node based interpolation.

See also:

[FIELD_ROUTINES::DofTypes](#),[FIELD_ROUTINES](#)

Definition at line 125 of file field_routines.f90.

Referenced by [FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET\(\)](#).

5.23.2.4 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_POINT_DOF_TYPE = 4

The dof is from a field variable component with point based interpolation.

See also:

[FIELD_ROUTINES::DofTypes](#),[FIELD_ROUTINES](#)

Definition at line 126 of file field_routines.f90.

5.24 FIELD_ROUTINES::ParameterSetTypes

Field parameter set type parameters.

Variables

- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_NUMBER_OF_SET_TYPES](#) = 99
The maximum number of different parameter sets for a field.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_VALUES_SET_TYPE](#) = 1
The parameter set corresponding to the field values.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_BOUNDARY_CONDITIONS_SET_TYPE](#) = 2
The parameter set corresponding to the field boundary conditions.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_INITIAL_CONDITIONS_SET_TYPE](#) = 3
The parameter set corresponding to the field initial conditions.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_ANALYTIC_SET_TYPE](#) = 4
The parameter set corresponding to the field values.

5.24.1 Detailed Description

Field parameter set type parameters.

[Todo](#)

make program defined constants negative?

See also:

[FIELD_ROUTINES](#)

5.24.2 Variable Documentation

5.24.2.1 INTEGER(INTG),parameter [FIELD_ROUTINES::FIELD_ANALYTIC_SET_TYPE](#) = 4

The parameter set corresponding to the field values.

See also:

[FIELD_ROUTINES::ParameterSetTypes](#),[FIELD_ROUTINES](#)

Definition at line 137 of file `field_routines.f90`.

Referenced by `ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_NODE_ANALYTIC_VALUE_GET()`, `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_PARAMETER_SET_UPDATE()`, and `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP()`.

**5.24.2.2 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_BOUNDARY_-
CONDITIONS_SET_TYPE = 2**

The parameter set corresponding to the field boundary conditions.

See also:

[FIELD_ROUTINES::ParameterSetTypes](#),[FIELD_ROUTINES](#)

Definition at line 135 of file field_routines.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_-SET_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_-PARAMETER_SET_UPDATE(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-EQUATIONS_SET_STANDARD_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_-ELASTICITY_EQUATIONS_SET_SETUP().

**5.24.2.3 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_INITIAL_CONDITIONS_-
SET_TYPE = 3**

The parameter set corresponding to the field initial conditions.

See also:

[FIELD_ROUTINES::ParameterSetTypes](#),[FIELD_ROUTINES](#)

Definition at line 136 of file field_routines.f90.

**5.24.2.4 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_NUMBER_OF_SET_TYPES
= 99**

The maximum number of different parameter sets for a field.

See also:

[FIELD_ROUTINES::ParameterSetTypes](#),[FIELD_ROUTINES](#)

Definition at line 133 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

5.24.2.5 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_VALUES_SET_TYPE = 1

The parameter set corresponding to the field values.

See also:

[FIELD_ROUTINES::ParameterSetTypes](#),[FIELD_ROUTINES](#)

Definition at line 134 of file field_routines.f90.

Referenced by ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_NODE_-
NUMERICIAL_VALUE_GET(), FIELD_ROUTINES::FIELD_CREATE_FINISH(), FIELD_-
ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), FIELD_IO_-
ROUTINES::FIELD_IO_CREATE_FIELDS(), FIELD_IO_ROUTINES::FIELD_IO_EXPORT_-
NODES_INTO_LOC(), FIELD_IO_ROUTINES::FIELD_IO_FILEDS_IMPORT(), LAPLACE_-
EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_-
SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATIONFINITE_ELEMENT_-
CALCULATE(), SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE(), and SOLVER_-
ROUTINES::SOLVER_VARIABLES_UPDATE().

5.25 FIELD_ROUTINES::ScalingTypes

Field scaling type parameters.

Variables

- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_NO_SCALING = 0](#)
The field is not scaled.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_UNIT_SCALING = 1](#)
The field has unit scaling.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_ARC_LENGTH_SCALING = 2](#)
The field has arc length scaling.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_ARITHMETIC_MEAN_SCALING = 3](#)
The field has arithmetic mean of the arc length scaling.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_HARMONIC_MEAN_SCALING = 4](#)
The field has geometric mean of the arc length scaling.

5.25.1 Detailed Description

Field scaling type parameters.

See also:

[FIELD_ROUTINES](#)

5.25.2 Variable Documentation

5.25.2.1 INTEGER(INTG),parameter [FIELD_ROUTINES::FIELD_ARC_LENGTH_SCALING = 2](#)

The field has arc length scaling.

See also:

[FIELD_ROUTINES::ScalingTypes](#),[FIELD_ROUTINES](#)

Definition at line 146 of file `field_routines.f90`.

Referenced by `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET()`, and `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`.

5.25.2.2 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_ARITHMETIC_MEAN_SCALING = 3

The field has arithmetic mean of the arc length scaling.

See also:

[FIELD_ROUTINES::ScalingTypes](#),[FIELD_ROUTINES](#)

Definition at line 147 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FIELD_CREATE_FINISH(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(), and FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

5.25.2.3 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_HARMONIC_MEAN_SCALING = 4

The field has geometric mean of the arc length scaling.

See also:

[FIELD_ROUTINES::ScalingTypes](#),[FIELD_ROUTINES](#)

Definition at line 148 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(), and FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

5.25.2.4 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_NO_SCALING = 0

The field is not scaled.

See also:

[FIELD_ROUTINES::ScalingTypes](#),[FIELD_ROUTINES](#)

Definition at line 144 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(), and FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

5.25.2.5 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_UNIT_SCALING = 1

The field has unit scaling.

See also:

[FIELD_ROUTINES::ScalingTypes](#),[FIELD_ROUTINES](#)

Definition at line 145 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(), and FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

5.26 GENERATED_MESH_ROUTINES::GeneratedMeshTypes

Generated mesh types.

Variables

- INTEGER(INTG), parameter [GENERATED_MESH_ROUTINES::GENERATED_MESH_REGULAR_MESH_TYPE = 1](#)
A regular generated mesh.
- INTEGER(INTG), parameter [GENERATED_MESH_ROUTINES::GENERATED_MESH_POLAR_MESH_TYPE = 2](#)
A polar generated mesh.
- INTEGER(INTG), parameter [GENERATED_MESH_ROUTINES::GENERATED_MESH_FRACTAL_TREE_MESH_TYPE = 3](#)
A fractal tree generated mesh.

5.26.1 Detailed Description

Generated mesh types.

See also:

[GENERATED_MESH_ROUTINES](#)

5.26.2 Variable Documentation

5.26.2.1 INTEGER(INTG),parameter [GENERATED_MESH_ROUTINES::GENERATED_MESH_FRACTAL_TREE_MESH_TYPE = 3](#)

A fractal tree generated mesh.

See also:

[GENERATED_MESH_ROUTINES::GeneratedMeshTypes](#),[GENERATED_MESH_ROUTINES](#)

Definition at line 70 of file generated_mesh_routines.f90.

5.26.2.2 INTEGER(INTG),parameter [GENERATED_MESH_ROUTINES::GENERATED_MESH_POLAR_MESH_TYPE = 2](#)

A polar generated mesh.

See also:

[GENERATED_MESH_ROUTINES::GeneratedMeshTypes](#),[GENERATED_MESH_ROUTINES](#)

Definition at line 69 of file generated_mesh_routines.f90.

5.26.2.3 INTEGER(INTG),parameter GENERATED_MESH_ROUTINES::GENERATED_MESH_REGULAR_MESH_TYPE = 1

A regular generated mesh.

See also:

[GENERATED_MESH_ROUTINES::GeneratedMeshTypes](#), [GENERATED_MESH_ROUTINES](#)

Definition at line 68 of file generated_mesh_routines.f90.

Referenced by
GENERATED_MESH_ROUTINES::GENERATED_MESH_BASIS_GET(),
GENERATED_MESH_ROUTINES::GENERATED_MESH_BASIS_SET_PTR(),
GENERATED_MESH_ROUTINES::GENERATED_MESH_CREATE_FINISH(),
GENERATED_MESH_ROUTINES::GENERATED_MESH_EXTENT_GET(),
GENERATED_MESH_ROUTINES::GENERATED_MESH_EXTENT_SET_PTR(),
GENERATED_MESH_ROUTINES::GENERATED_MESH_NUMBER_OF_ELEMENTS_GET(),
GENERATED_MESH_ROUTINES::GENERATED_MESH_NUMBER_OF_ELEMENTS_SET_PTR(),
GENERATED_MESH_ROUTINES::GENERATED_MESH_ORIGIN_GET(),
GENERATED_MESH_ROUTINES::GENERATED_MESH_ORIGIN_SET_PTR(),
GENERATED_MESH_ROUTINES::GENERATED_MESH_REGULAR_INITIALISE(), and
GENERATED_MESH_ROUTINES::GENERATED_MESH_TYPE_SET_PTR().

5.27 INPUT_OUTPUT::MatrixNameIndexFormat

Output type parameter.

Variables

- INTEGER(INTG), parameter [INPUT_OUTPUT::WRITE_STRING_MATRIX_NAME_ONLY = 1](#)

Write the matrix name with out any indices.

- INTEGER(INTG), parameter [INPUT_OUTPUT::WRITE_STRING_MATRIX_NAME_AND_INDICES = 2](#)

Write the matrix name together with the matrix indices.

5.27.1 Detailed Description

Output type parameter.

See also:

[INPUT_OUTPUT](#)

5.27.2 Variable Documentation

5.27.2.1 INTEGER(INTG),parameter INPUT_OUTPUT::WRITE_STRING_MATRIX_NAME_AND_INDICES = 2

Write the matrix name together with the matrix indices.

See also:

[INPUT_OUTPUT::MatrixNameIndexFormat](#), [INPUT_OUTPUT::MatrixNameIndexFormat](#)

Definition at line 63 of file input_output.f90.

Referenced by COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE(), MATRIX_VECTOR::MATRIX_OUTPUT(), INPUT_OUTPUT::WRITE_STRING_MATRIX_DP(), INPUT_OUTPUT::WRITE_STRING_MATRIX_INTG(), INPUT_OUTPUT::WRITE_STRING_MATRIX_L(), INPUT_OUTPUT::WRITE_STRING_MATRIX_LINTG(), and INPUT_OUTPUT::WRITE_STRING_MATRIX_SP().

5.27.2.2 INTEGER(INTG),parameter INPUT_OUTPUT::WRITE_STRING_MATRIX_NAME_ONLY = 1

Write the matrix name with out any indices.

See also:

[INPUT_OUTPUT::MatrixNameIndexFormat](#), [INPUT_OUTPUT::MatrixNameIndexFormat](#)

Definition at line 62 of file input_output.f90.

Referenced by INPUT_OUTPUT::WRITE_STRING_MATRIX_DPO(), INPUT_OUTPUT::WRITE_STRING_MATRIX_INTG(), INPUT_OUTPUT::WRITE_STRING_MATRIX_L(), INPUT_OUTPUT::WRITE_STRING_MATRIX_LINTG(), and INPUT_OUTPUT::WRITE_STRING_MATRIX_SP().

5.28 KINDS::IntegerKinds

Kind parameters for integer data types.

Variables

- INTEGER, parameter [KINDS::INTG](#) = SELECTED_INT_KIND(9)
Standard integer kind.
- INTEGER, parameter [KINDS::SINTG](#) = SELECTED_INT_KIND(4)
Short integer kind.
- INTEGER, parameter [KINDS::LINTG](#) = SELECTED_INT_KIND(18)
Long integer kind.
- INTEGER, parameter [KINDS::PTR](#) = INTG
Pointer integer kind.

5.28.1 Detailed Description

Kind parameters for integer data types.

See also:

[KINDS](#)

5.28.2 Variable Documentation

5.28.2.1 INTEGER,parameter KINDS::INTG = SELECTED_INT_KIND(9)

Standard integer kind.

See also:

[KINDS::IntegerKinds](#), [KINDS](#)

Definition at line 54 of file kinds.f90.

Referenced by MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE(), MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(), and BINARY_-FILE::OPEN_CMISS_BINARY_FILE().

5.28.2.2 INTEGER,parameter KINDS::LINTG = SELECTED_INT_KIND(18)

Long integer kind.

See also:

[KINDS::IntegerKinds](#), [KINDS](#)

Definition at line 56 of file kinds.f90.

5.28.2.3 INTEGER,parameter KINDS::PTR = INTG

Pointer integer kind.

Definition at line 57 of file kinds.f90.

```
Referenced      by      COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_
FINISH(),      COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_START(),
COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_NUMBER(),
COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_PTR(),      COORDINATE-
ROUTINES::COORDINATE_SYSTEM_USER_NUMBER_FIND(),      COORDINATE-
ROUTINES::COORDINATE_SYSTEMS_FINALISE(), COORDINATE_ROUTINES::COORDINATE-
SYSTEMS_INITIALISE(),      MESH_ROUTINES::DECOMPOSITION_CREATE-
FINISH(),      MESH_ROUTINES::DECOMPOSITION_CREATE_START(),      MESH-
ROUTINES::DECOMPOSITION_DESTROY(),      MESH_ROUTINES::DECOMPOSITION-
ELEMENT_DOMAIN_CALCULATE(),      MESH_ROUTINES::DECOMPOSITION_ELEMENT-
DOMAIN_GET(),      MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_SET(),
MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(),      DOMAIN-
MAPPINGS::DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE(),      MESH-
ROUTINES::DOMAIN_TOPOLOGY_INITIALISE_FROM_MESH(),      EQUATIONS-
MAPPING_ROUTINES::EQUATIONS_MAPPING_CALCULATE(),      EQUATIONS_MAPPING-
ROUTINES::EQUATIONS_MAPPING_CREATE_VALUES_CACHE_INITIALISE(), EQUATIONS-
MAPPING_ROUTINES::EQUATIONS_MAPPING_LINEAR_MATRICES_VARIABLE-
TYPES_SET(),      EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_RESIDUAL-
VARIABLE_TYPE_SET(),      EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING-
RHS_VARIABLE_TYPE_SET(),      EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING-
MAPPING_SOURCE_VARIABLE_TYPE_SET(),      FIELD_ROUTINES::FIELD_CREATE_FINISH(),
FIELD_ROUTINES::FIELD_DESTROY(),      FIELD_ROUTINES::FIELD_INTERPOLATE-
GAUSS(),      FIELD_ROUTINES::FIELD_INTERPOLATE_XI(),      FIELD_ROUTINES::FIELD-
INTERPOLATION_PARAMETERS_ELEMENT_GET(),      FIELD_ROUTINES::FIELD-
INTERPOLATION_PARAMETERS_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATION-
PARAMETERS_LINE_GET(),      FIELD_IO_ROUTINES::FIELD_IO_CREATE_FIELDS(),
FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_ATTACH_LOCAL_PROCESS(),
FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_FINALIZE(),      FIELD_IO-
ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_INITIALISE(), FIELD_IO_ROUTINES::FIELD-
IO_ELEMENTAL_INFO_SET_SORT(),      FIELD_IO_ROUTINES::FIELD_IO_EXPORT-
ELEMENTAL_GROUP_HEADER_FORTRAN(),      FIELD_IO_ROUTINES::FIELD_IO-
EXPORT_ELEMENTS_INTO_(),      FIELD_IO_ROUTINES::FIELD_IO_EXPORT_NODAL-
GROUP_HEADER_FORTRA(),      FIELD_IO_ROUTINES::FIELD_IO_EXPORT_NODES-
INTO_LOC(),      FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH(),      FIELD_IO-
ROUTINES::FIELD_IO_NODAL_INFO_SET_ATTACH_LOCAL_PROCESS(),      FIELD_IO-
ROUTINES::FIELD_IO_NODAL_INFO_SET_FINALIZE(),      FIELD_IO_ROUTINES::FIELD-
IO_NODAL_INFO_SET_INITIALISE(),      FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO-
SET_SORT(),      FIELD_ROUTINES::FIELD_VARIABLE_COMPONENT_INITIALISE(), FIELD-
ROUTINES::FIELDS_FINALISE(),      FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY-
FINITE_ELEMENT_RESIDUAL_EVALUATE(),      FINITE_ELASTICITY_ROUTINES::FINITE-
ELASTICITY_GAUSS_CAUCHY_TENSOR(),      FINITE_ELASTICITY_ROUTINES::FINITE-
ELASTICITY_GAUSS_DFDZ(),      FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY-
PROBLEM_SETUP(),      GENERATED_MESH_ROUTINES::GENERATED_MESH_CREATE-
START(),      GENERATED_MESH_ROUTINES::GENERATED_MESH_DESTROY_NUMBER(),
GENERATED_MESH_ROUTINES::GENERATED_MESH_DESTROY_PTR(),      GENERATED-
MESH_ROUTINES::GENERATED_MESH_USER_NUMBER_FIND(),      GENERATED-
MESH_ROUTINES::GENERATED_MESHES_FINALISE(),      LAPLACE_EQUATIONS-
ROUTINES::LAPLACE_EQUATIONFINITE_ELEMENT_CALCULATE(),      LAPLACE-
EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(),
```

LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SETUP(), MESH_ROUTINES::MESH_CREATE_FINISH(), MESH_ROUTINES::MESH_CREATE_START(), MESH_ROUTINES::MESH_DESTROY(), MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET_PTR(), MESH_ROUTINES::MESH_NUMBER_OF_ELEMENTS_SET_PTR(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_FINISH(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_START(), MESH_ROUTINES::MESH_TOPOLOGY_FINALISE(), MESH_ROUTINES::MESH_TOPOLOGY_INITIALISE(), MESH_ROUTINES::MESH_USER_NUMBER_FIND(), MESH_ROUTINES::MESHS_FINALISE(), PROBLEM_ROUTINES::PROBLEM_CREATE_FINISH(), PROBLEM_ROUTINES::PROBLEM_CREATE_START(), PROBLEM_ROUTINES::PROBLEM_DESTROY_NUMBER(), PROBLEM_ROUTINES::PROBLEM_DESTROY_PTR(), PROBLEM_ROUTINES::PROBLEM_SOLUTION_EQUATIONS_SET_ADD(), PROBLEM_ROUTINES::PROBLEM_SOLUTION_JACOBIAN_EVALUATE(), PROBLEM_ROUTINES::PROBLEM_SOLUTION_RESIDUAL_EVALUATE(), PROBLEM_ROUTINES::PROBLEM_SOLUTION_SOLVE(), PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_CREATE_FINISH(), PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_FINALISE(), PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_INITIALISE(), PROBLEM_ROUTINES::PROBLEM_SOLVE(), PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_START(), PROBLEM_ROUTINES::PROBLEM_SOLVER_DESTROY(), PROBLEM_ROUTINES::PROBLEM_SOLVER_GET(), PROBLEM_ROUTINES::PROBLEM_USER_NUMBER_FIND(), PROBLEM_ROUTINES::PROBLEMS_FINALISE(), REGION_ROUTINES::REGION_CREATE_FINISH(), REGION_ROUTINES::REGION_CREATE_START(), REGION_ROUTINES::REGION_DESTROY(), REGION_ROUTINES::REGION_SUB_REGION_CREATE_FINISH(), REGION_ROUTINES::REGION_SUB_REGION_CREATE_START(), REGION_ROUTINES::REGION_USER_NUMBER_FIND(), REGION_ROUTINES::REGION_USER_NUMBER_FIND_PTR(), REGION_ROUTINES::REGIONS_FINALISE(), SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_SOLVE(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE(), SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_FINISH(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_FINALISE(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_INITIALISE(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_OUTPUT(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_GET(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_INITIALISE(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_CALCULATE(), SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_SOLVE(), SOLVER_ROUTINES::SOLVER_NONLINEAR_RESIDUAL_EVALUATE(), and SOLVER_ROUTINES::SOLVER_VARIABLES_UPDATE().

5.28.2.4 INTEGER,parameter KINDS::SINTG = SELECTED_INT_KIND(4)

Short integer kind.

See also:

[KINDS::IntegerKinds](#), [KINDS](#)

Definition at line 55 of file kinds.f90.

5.29 KINDS::RealKinds

Kind parameters for real data types.

Variables

- INTEGER, parameter **KINDS::SP** = SELECTED_REAL_KIND(6)
Single precision real kind.
- INTEGER, parameter **KINDS::DP** = SELECTED_REAL_KIND(15)
Double precision real kind.
- INTEGER, parameter **KINDS::QP** = SELECTED_REAL_KIND(30)
Quadruple precision real kind.

5.29.1 Detailed Description

Kind parameters for real data types.

See also:

[KINDS](#)

5.29.2 Variable Documentation

5.29.2.1 INTEGER,parameter KINDS::DP = SELECTED_REAL_KIND(15

Double precision real kind.

See also:

[KINDS::RealKinds](#), [KINDS](#)

Definition at line 65 of file kinds.f90.

5.29.2.2 INTEGER,parameter KINDS::QP = SELECTED_REAL_KIND(30

Quadruple precision real kind.

See also:

[KINDS::RealKinds](#), [KINDS](#)

Definition at line 66 of file kinds.f90.

5.29.2.3 INTEGER,parameter KINDS::SP = SELECTED_REAL_KIND(6)

Single precision real kind.

See also:

[KINDS::RealKinds](#),[KINDS](#)

Definition at line 64 of file kinds.f90.

5.30 KINDS::ComplexKinds

Kind parameters for complex data types.

Variables

- INTEGER, parameter [KINDS::SPC](#) = KIND((1.0_SP)
- INTEGER, parameter [KINDS::_SP](#)
Single precision complex kind.
- INTEGER, parameter [KINDS::DPC](#) = KIND((1.0_DP)
- INTEGER, parameter [KINDS::_DP](#)
Double precision complex kind.

5.30.1 Detailed Description

Kind parameters for complex data types.

See also:

[KINDS](#)

5.30.2 Variable Documentation

5.30.2.1 INTEGER,parameter KINDS::_DP

Double precision complex kind.

See also:

[KINDS::ComplexKinds](#), [KINDS](#)

Definition at line 74 of file kinds.f90.

Referenced by ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_CALCULATE(), ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_INTEGRAL_ANALYTIC_-
VALUE_GET(), ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_INTEGRAL_-
NUMERICAL_VALUE_GET(), ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_INTEGRAL_-
PERCENT_ERROR_GET(), ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_INTEGRAL_RELATIVE_ERROR_GET(), ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_NODE_PERCENT_ERROR_GET(), ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_NODE_RELATIVE_ERROR_GET(), ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_RMS_ABSOLUTE_-
ERROR_GET(), ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_RMS_-
PERCENT_ERROR_GET(), ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_RMS_RELATIVE_ERROR_GET(), BASE_ROUTINES::BASE_ROUTINES_INITIALISE(), COORDINATE_ROUTINES::CO(), COORDINATE_ROUTINES::COORDINATE_CONVERT_-
FROM_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_DP(),

COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE_DP(), COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_NORM(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_START(), COORDINATE_ROUTINES::COORDINATE_SYSTEMS_INITIALISE(), COORDINATE_ROUTINES::D2ZX_DP(), MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(), MATHS::DETERMINANT_FULL_DP(), COORDINATE_ROUTINES::DXZ_DP(), COORDINATE_ROUTINES::DZX_DP(), MATHS::EDP_DP(), MATHS::EIGENVALUE_FULL_DP(), MATHS::EIGENVECTOR_FULL_DP(), MATHS::EIGENVECTOR_FULL_SP(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_CREATE_VALUES_CACHE_INITIALISE(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_EQUIV_MATRIX_TO_VARIABLE_MAP_INITIALISE(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_NONLINEAR_MAPPING_INITIALISE(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_RHS_MAPPING_INITIALISE(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_SOURCE_MAPPING_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITYFINITE_ELEMENT_RESIDUAL_EVALUATE(), FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_GAUSS_CAUCHY_TENSOR(), FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_GAUSS_DEFORMATION_GRADEINT_TENSOR(), FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_GAUSS_DFDZ(), FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_GAUSS_DXDNU(), GENERATED_MESH_ROUTINES::GENERATED_MESH_REGULAR_CREATE_FINISH(), MATHS::I0_DP(), MATHS::I1_DP(), MATHS::INVERT_FULL_DP(), MATHS::INVERT_FULL_SP(), MATHS::K0_DP(), MATHS::K1_DP(), MATHS::KDP_DP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATIONFINITE_ELEMENT_CALCULATE(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_INTE(), MATHS::MATRIX_PRODUCT_DP(), MATRIX_VECTOR::MATRIX_VALUES_GET_DP(), MATRIX_VECTOR::MATRIX_VALUES_GET_DP1(), MATRIX_VECTOR::MATRIX_VALUES_GET_DP2(), NODE_ROUTINES::NODES_CREATE_START(), STRINGS::NUMBER_TO_CHARACTER_DP(), STRINGS::NUMBER_TO_VSTRING_DP(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_INITIALISE(), SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE(), SOLVER_ROUTINES::SOLVER_NONLINEAR_INITIALISE(), and SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_INITIALISE().

5.30.2.2 INTEGER,parameter KINDS::SP

Single precision complex kind.

See also:

[KINDS::ComplexKinds,KINDS](#)

Definition at line 73 of file kinds.f90.

Referenced by BASE_ROUTINES::BASE_ROUTINES_INITIALISE(), COORDINATE_ROUTINES::CO(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_SP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_SP(), MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE(), MATHS::DETERMINANT_FULL_SP(), MATHS::EDP_SP(), MATHS::EIGENVALUE_FULL_SP(), MATHS::EIGENVECTOR_FULL_SP(), BASE_ROUTINES::ENTERS(), MATHS::I0_SP(), MATHS::I1_SP(), MATHS::INVERT_FULL_SP(), MATHS::K0_SP(), MATHS::K1_SP(), MATHS::KDP_SP(), MATHS::MATRIX_PRODUCT_SP(), MATRIX_VECTOR::MATRIX_VALUES_GET_SP(), MATRIX_VECTOR::MATRIX_VALUES_GET_SP1(), MATRIX_VECTOR::MATRIX_VALUES_GET_SP2()

VALUES_GET_SP2(), STRINGS::NUMBER_TO_CHARACTER_SP(), STRINGS::NUMBER_TO_VSTRING_SP(), and BASE_ROUTINES::TIMING_SET_ON().

5.30.2.3 INTEGER,parameter KINDS::DPC = KIND((1.0_DP

Definition at line 74 of file kinds.f90.

5.30.2.4 INTEGER,parameter KINDS::SPC = KIND((1.0_SP

Definition at line 73 of file kinds.f90.

5.31 LISTS::DataType

Data type parameters for a list.

Variables

- INTEGER(INTG), parameter [LISTS::LIST_INTG_TYPE = INTEGER_TYPE](#)
Integer data type for a list.
- INTEGER(INTG), parameter [LISTS::LIST_SP_TYPE = SINGLE_REAL_TYPE](#)
Single precision real data type for a list.
- INTEGER(INTG), parameter [LISTS::LIST_DP_TYPE = DOUBLE_REAL_TYPE](#)
Double precision real data type for a list.

5.31.1 Detailed Description

Data type parameters for a list.

See also:

[LISTS](#)

5.31.2 Variable Documentation

5.31.2.1 INTEGER(INTG),parameter [LISTS::LIST_DP_TYPE = DOUBLE_REAL_TYPE](#)

Double precision real data type for a list.

See also:

[LISTS::DataType](#),[LISTS](#)

Definition at line 65 of file lists.f90.

Referenced by [LISTS::LIST_CREATE_FINISH\(\)](#), [LISTS::LIST_DATA_TYPE_SET\(\)](#), [LISTS::LIST_DETACH_AND_DESTROY_DP\(\)](#), [LISTS::LIST_ITEM_ADD_DP1\(\)](#), [LISTS::LIST_ITEM_DELETE\(\)](#), [LISTS::LIST_ITEM_IN_LIST_DP1\(\)](#), and [LISTS::LIST_REMOVE_DUPLICATES\(\)](#).

5.31.2.2 INTEGER(INTG),parameter [LISTS::LIST_INTG_TYPE = INTEGER_TYPE](#)

Integer data type for a list.

See also:

[LISTS::DataType](#),[LISTS](#)

Definition at line 63 of file lists.f90.

Referenced by [MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET\(\)](#), [DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE\(\)](#),

FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), LISTS::LIST_CREATE_FINISH(), LISTS::LIST_CREATE_START(), LISTS::LIST_DATA_TYPE_SET(), LISTS::LIST_DETACH_AND_DESTROY_INTG(), LISTS::LIST_ITEM_ADD_INTG1(), LISTS::LIST_ITEM_DELETE(), LISTS::LIST_ITEM_IN_LIST_INTG1(), LISTS::LIST_REMOVE_DUPLICATES(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE(), and SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_CALCULATE().

5.31.2.3 INTEGER(INTG),parameter LISTS::LIST_SP_TYPE = SINGLE_REAL_TYPE

Single precision real data type for a list.

See also:

[LISTS::DataType](#),[LISTS](#)

Definition at line 64 of file lists.f90.

Referenced by LISTS::LIST_CREATE_FINISH(), LISTS::LIST_DATA_TYPE_SET(), LISTS::LIST_DETACH_AND_DESTROY_SP(), LISTS::LIST_ITEM_ADD_SP1(), LISTS::LIST_ITEM_DELETE(), LISTS::LIST_ITEM_IN_LIST_SP1(), and LISTS::LIST_REMOVE_DUPLICATES().

5.32 LISTS::SortingOrder

Sorting order parameters for a list.

Variables

- INTEGER(INTG), parameter [LISTS::LIST_UNSORTED_TYPE = 1](#)
Unsorted list type.
- INTEGER(INTG), parameter [LISTS::LIST_SORT_ASCENDING_TYPE = 2](#)
Ascending order for sort.
- INTEGER(INTG), parameter [LISTS::LIST_SORT_DESCENDING_TYPE = 3](#)
Descending order for sort.

5.32.1 Detailed Description

Sorting order parameters for a list.

See also:

[LISTS](#)

5.32.2 Variable Documentation

5.32.2.1 INTEGER(INTG),parameter [LISTS::LIST_SORT_ASCENDING_TYPE = 2](#)

Ascending order for sort.

See also:

[LISTS::SortingOrder](#),[LISTS](#)

Definition at line 73 of file lists.f90.

Referenced by [LISTS::LIST_CREATE_START\(\)](#).

5.32.2.2 INTEGER(INTG),parameter [LISTS::LIST_SORT_DESCENDING_TYPE = 3](#)

Descending order for sort.

See also:

[LISTS::SortingOrder](#),[LISTS](#)

Definition at line 74 of file lists.f90.

5.32.2.3 INTEGER(INTG),parameter LISTS::LIST_UNSORTED_TYPE = 1

Unsorted list type.

See also:

[LISTS::SortingOrder](#),[LISTS](#)

Definition at line 72 of file lists.f90.

5.33 LISTS::SortingMethod

Sorting method parameters for a list.

Variables

- INTEGER(INTG), parameter [LISTS::LIST_BUBBLE_SORT_METHOD = 1](#)
Bubble sort method.
- INTEGER(INTG), parameter [LISTS::LIST_SHELL_SORT_METHOD = 2](#)
Shell sort method.
- INTEGER(INTG), parameter [LISTS::LIST_HEAP_SORT_METHOD = 3](#)
Heap sort method.

5.33.1 Detailed Description

Sorting method parameters for a list.

See also:

[LISTS](#)

5.33.2 Variable Documentation

5.33.2.1 INTEGER(INTG),parameter [LISTS::LIST_BUBBLE_SORT_METHOD = 1](#)

Bubble sort method.

See also:

[LISTS_SortingMethod](#),[LISTS](#)

Definition at line 81 of file lists.f90.

5.33.2.2 INTEGER(INTG),parameter [LISTS::LIST_HEAP_SORT_METHOD = 3](#)

Heap sort method.

See also:

[LISTS_SortingMethod](#),[LISTS](#)

Definition at line 83 of file lists.f90.

Referenced by [LISTS::LIST_CREATE_START\(\)](#).

5.33.2.3 INTEGER(INTG),parameter LISTS::LIST_SHELL_SORT_METHOD = 2

Shell sort method.

See also:

LISTS_SortingMethod, [LISTS](#)

Definition at line 82 of file lists.f90.

5.34 MATRIX_VECTOR::DataTypes

Matrix vector data types.

Variables

- INTEGER(INTG), parameter **MATRIX_VECTOR::MATRIX_VECTOR_INTG_TYPE** = INTEGER_TYPE
Integer matrix-vector data type.
- INTEGER(INTG), parameter **MATRIX_VECTOR::MATRIX_VECTOR_SP_TYPE** = SINGLE_REAL_TYPE
Single precision real matrix-vector data type.
- INTEGER(INTG), parameter **MATRIX_VECTOR::MATRIX_VECTOR_DP_TYPE** = DOUBLE_REAL_TYPE
Double precision real matrix-vector data type.
- INTEGER(INTG), parameter **MATRIX_VECTOR::MATRIX_VECTOR_L_TYPE** = LOGICAL_TYPE
Logical matrix-vector data type.

5.34.1 Detailed Description

Matrix vector data types.

See also:

[MATRIX_VECTOR](#)

5.34.2 Variable Documentation

5.34.2.1 INTEGER(INTG),parameter **MATRIX_VECTOR::MATRIX_VECTOR_DP_TYPE** = DOUBLE_REAL_TYPE

Double precision real matrix-vector data type.

See also:

[MATRIX_VECTOR::DataTypes](#), [MATRIX_VECTOR](#)

Definition at line 151 of file matrix_vector.f90.

Referenced by FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_DP(), MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_CREATE_START(), MATRIX_VECTOR::MATRIX_DATA_GET_DP(), MATRIX_VECTOR::MATRIX_DATA_TYPE_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_VALUES_ADD_DP(), MATRIX_VECTOR::MATRIX_VALUES_ADD_DP1(), MATRIX_VECTOR::MATRIX_VALUES_ADD_DP2(), MATRIX_VECTOR::MATRIX_VALUES_GET_DP(), MATRIX_VECTOR::MATRIX_VALUES_GET_DP1(), MATRIX_VECTOR::MATRIX_VALUES_GET_DP2(),

MATRIX_VECTOR::MATRIX_VALUES_SET_DP(), MATRIX_VECTOR::MATRIX_VALUES_SET_DP1(), MATRIX_VECTOR::MATRIX_VALUES_SET_DP2(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_FINISH(), MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_DP(), MATRIX_VECTOR::VECTOR_CREATE_FINISH(), MATRIX_VECTOR::VECTOR_CREATE_START(), MATRIX_VECTOR::VECTOR_DATA_GET_DP(), MATRIX_VECTOR::VECTOR_DATA_TYPE_SET(), MATRIX_VECTOR::VECTOR_VALUES_GET_DP(), MATRIX_VECTOR::VECTOR_VALUES_GET_DP1(), MATRIX_VECTOR::VECTOR_VALUES_SET_DP(), and MATRIX_VECTOR::VECTOR_VALUES_SET_DP1().

5.34.2.2 INTEGER(INTG),parameter MATRIX_VECTOR::MATRIX_VECTOR_INTG_TYPE = INTEGER_TYPE

Integer matrix-vector data type.

See also:

[MATRIX_VECTOR::DataTypes](#), [MATRIX_VECTOR](#)

Definition at line 149 of file matrix_vector.f90.

Referenced by MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_INTG(), MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_DATA_GET_INTG(), MATRIX_VECTOR::MATRIX_DATA_TYPE_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG(), MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG1(), MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG2(), MATRIX_VECTOR::MATRIX_VALUES_GET_INTG(), MATRIX_VECTOR::MATRIX_VALUES_GET_INTG1(), MATRIX_VECTOR::MATRIX_VALUES_GET_INTG2(), MATRIX_VECTOR::MATRIX_VALUES_SET_INTG(), MATRIX_VECTOR::MATRIX_VALUES_SET_INTG1(), MATRIX_VECTOR::MATRIX_VALUES_SET_INTG2(), MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_INTG(), MATRIX_VECTOR::VECTOR_CREATE_FINISH(), MATRIX_VECTOR::VECTOR_DATA_GET_INTG(), MATRIX_VECTOR::VECTOR_DATA_TYPE_SET(), MATRIX_VECTOR::VECTOR_VALUES_GET_INTG(), MATRIX_VECTOR::VECTOR_VALUES_GET_INTG1(), MATRIX_VECTOR::VECTOR_VALUES_SET_INTG(), and MATRIX_VECTOR::VECTOR_VALUES_SET_INTG1().

5.34.2.3 INTEGER(INTG),parameter MATRIX_VECTOR::MATRIX_VECTOR_L_TYPE = LOGICAL_TYPE

Logical matrix-vector data type.

See also:

[MATRIX_VECTOR::DataTypes](#), [MATRIX_VECTOR](#)

Definition at line 152 of file matrix_vector.f90.

Referenced by MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_L(), MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_DATA_GET_L(), MATRIX_VECTOR::MATRIX_DATA_TYPE_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_VALUES_ADD_L(), MATRIX_VECTOR::MATRIX_VALUES_ADD_L1(), MATRIX_VECTOR::MATRIX_VALUES_ADD_L2(), MATRIX_VECTOR::MATRIX_VALUES_GET_L(), and MATRIX_VECTOR::MATRIX_VALUES_GET_L1().

MATRIX_VECTOR::MATRIX_VALUES_GET_L2(), MATRIX_VECTOR::MATRIX_VALUES_SET_L(), MATRIX_VECTOR::MATRIX_VALUES_SET_L1(), MATRIX_VECTOR::MATRIX_VALUES_SET_L2(), MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_L(), MATRIX_VECTOR::VECTOR_CREATE_FINISH(), MATRIX_VECTOR::VECTOR_DATA_GET_L(), MATRIX_VECTOR::VECTOR_DATA_TYPE_SET(), MATRIX_VECTOR::VECTOR_VALUES_GET_L(), MATRIX_VECTOR::VECTOR_VALUES_GET_L1(), MATRIX_VECTOR::VECTOR_VALUES_SET_L(), and MATRIX_VECTOR::VECTOR_VALUES_SET_L1().

5.34.2.4 INTEGER(INTG),parameter MATRIX_VECTOR::MATRIX_VECTOR_SP_TYPE = SINGLE_REAL_TYPE

Single precision real matrix-vector data type.

See also:

[MATRIX_VECTOR::DataTypes](#), [MATRIX_VECTOR](#)

Definition at line 150 of file matrix_vector.f90.

Referenced by MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_SP(), MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_DATA_GET_SP(), MATRIX_VECTOR::MATRIX_DATA_TYPE_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_VALUES_ADD_SP(), MATRIX_VECTOR::MATRIX_VALUES_ADD_SP1(), MATRIX_VECTOR::MATRIX_VALUES_ADD_SP2(), MATRIX_VECTOR::MATRIX_VALUES_GET_SP(), MATRIX_VECTOR::MATRIX_VALUES_GET_SP10(), MATRIX_VECTOR::MATRIX_VALUES_GET_SP2(), MATRIX_VECTOR::MATRIX_VALUES_SET_SP(), MATRIX_VECTOR::MATRIX_VALUES_SET_SP10(), MATRIX_VECTOR::MATRIX_VALUES_SET_SP2(), MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_SP(), MATRIX_VECTOR::VECTOR_CREATE_FINISH(), MATRIX_VECTOR::VECTOR_DATA_GET_SP(), MATRIX_VECTOR::VECTOR_DATA_TYPE_SET(), MATRIX_VECTOR::VECTOR_VALUES_GET_SP(), MATRIX_VECTOR::VECTOR_VALUES_GET_SP1(), MATRIX_VECTOR::VECTOR_VALUES_SET_SP(), and MATRIX_VECTOR::VECTOR_VALUES_SET_SP1().

5.35 MATRIX_VECTOR::StorageTypes

Matrix-vector storage type parameters.

Variables

- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_BLOCK_STORAGE_TYPE = 0`
Matrix block storage type.
- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_DIAGONAL_STORAGE_TYPE = 1`
Matrix diagonal storage type.
- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_COLUMN_MAJOR_STORAGE_TYPE = 2`
Matrix column major storage type.
- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_ROW_MAJOR_STORAGE_TYPE = 3`
Matrix row major storage type.
- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_COMPRESSED_ROW_STORAGE_TYPE = 4`
Matrix compressed row storage type.
- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE = 5`
Matrix compressed column storage type.
- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_ROW_COLUMN_STORAGE_TYPE = 6`
Matrix row-column storage type.

5.35.1 Detailed Description

Matrix-vector storage type parameters.

See also:

`MATRIX_VECTOR_MatrixStorageStructures`, `MATRIX_VECTOR`

5.35.2 Variable Documentation

5.35.2.1 INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_BLOCK_STORAGE_TYPE = 0`

Matrix block storage type.

See also:

[MATRIX_VECTOR::StorageTypes](#), [MATRIX_VECTOR](#)

Definition at line 159 of file matrix_vector.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUIVATIONS_SET_SETUP(), MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_CREATE_START(), MATRIX_VECTOR::MATRIX_DUPLICATE(), MATRIX_VECTOR::MATRIX_NUMBER_NON_ZEROS_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATION_FIND(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_GET(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_SET(), MATRIX_VECTOR::MATRIX_STORAGE_TYPE_SET(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_FINISH(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET(), and SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_INITIALISE().

5.35.2.2 INTEGER(INTG),parameter MATRIX_VECTOR::MATRIX_COLUMN_MAJOR_STORAGE_TYPE = 2

Matrix column major storage type.

See also:

[MATRIX_VECTOR::StorageTypes](#), [MATRIX_VECTOR](#)

Definition at line 161 of file matrix_vector.f90.

Referenced by MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_DUPLICATE(), MATRIX_VECTOR::MATRIX_NUMBER_NON_ZEROS_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATION_FIND(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_GET(), MATRIX_VECTOR::MATRIX_STORAGE_TYPE_SET(), and SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET().

5.35.2.3 INTEGER(INTG),parameter MATRIX_VECTOR::MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE = 5

Matrix compressed column storage type.

See also:

[MATRIX_VECTOR::StorageTypes](#), [MATRIX_VECTOR](#)

Definition at line 164 of file matrix_vector.f90.

Referenced by MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_DUPLICATE(), MATRIX_VECTOR::MATRIX_NUMBER_NON_ZEROS_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATION_FIND(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_GET(), MATRIX_VECTOR::MATRIX_STORAGE_TYPE_SET(), and SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET().

**5.35.2.4 INTEGER(INTG),parameter MATRIX_VECTOR::MATRIX_COMPRESSED_ROW -
STORAGE_TYPE = 4**

Matrix compressed row storage type.

See also:

[MATRIX_VECTOR::StorageTypes](#), [MATRIX_VECTOR](#)

Definition at line 163 of file matrix_vector.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUIVATIONS_SET_SETUP(), MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_DUPLICATE(), MATRIX_VECTOR::MATRIX_NUMBER_NON_ZEROS_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATION_FIND(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_GET(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_SET(), MATRIX_VECTOR::MATRIX_STORAGE_TYPE_SET(), and SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET().

**5.35.2.5 INTEGER(INTG),parameter MATRIX_VECTOR::MATRIX_DIAGONAL -
STORAGE_TYPE = 1**

Matrix diagonal storage type.

See also:

[MATRIX_VECTOR::StorageTypes](#), [MATRIX_VECTOR](#)

Definition at line 160 of file matrix_vector.f90.

Referenced by MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_DUPLICATE(), MATRIX_VECTOR::MATRIX_NUMBER_NON_ZEROS_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATION_FIND(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_GET(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_SET(), MATRIX_VECTOR::MATRIX_STORAGE_TYPE_SET(), and SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET().

**5.35.2.6 INTEGER(INTG),parameter MATRIX_VECTOR::MATRIX_ROW_COLUMN -
STORAGE_TYPE = 6**

Matrix row-column storage type.

See also:

[MATRIX_VECTOR::StorageTypes](#), [MATRIX_VECTOR](#)

Definition at line 165 of file matrix_vector.f90.

Referenced by MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_DUPLICATE(), MATRIX_VECTOR::MATRIX_NUMBER_NON_ZEROS_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATION_FIND(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_GET(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_SET(), MATRIX_VECTOR::MATRIX_STORAGE_TYPE_SET(), and SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET().

**5.35.2.7 INTEGER(INTG),parameter MATRIX_VECTOR::MATRIX_ROW_MAJOR_-
STORAGE_TYPE = 3**

Matrix row major storage type.

See also:

[MATRIX_VECTOR::StorageTypes](#),[MATRIX_VECTOR](#)

Definition at line 162 of file matrix_vector.f90.

Referenced by `MATRIX_VECTOR::MATRIX_CREATE_FINISH()`, `MATRIX_VECTOR::MATRIX_DUPLICATE()`, `MATRIX_VECTOR::MATRIX_NUMBER_NON_ZEROS_SET()`, `MATRIX_VECTOR::MATRIX_OUTPUT()`, `MATRIX_VECTOR::MATRIX_STORAGE_LOCATION_FIND()`, `MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_GET()`, `MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_SET()`, `MATRIX_VECTOR::MATRIX_STORAGE_TYPE_SET()`, and `SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET()`.

5.36 MESH_ROUTINES::DecompositionTypes

The Decomposition types parameters.

Variables

- INTEGER(INTG), parameter [MESH_ROUTINES::DECOMPOSITION_ALL_TYPE = 1](#)
The decomposition contains all elements.
- INTEGER(INTG), parameter [MESH_ROUTINES::DECOMPOSITION_CALCULATED_TYPE = 2](#)
The element decomposition is calculated by graph partitioning.
- INTEGER(INTG), parameter [MESH_ROUTINES::DECOMPOSITION_USER_DEFINED_TYPE = 3](#)
The user will set the element decomposition.

5.36.1 Detailed Description

The Decomposition types parameters.

See also:

[MESH_ROUTINES](#)

5.36.2 Variable Documentation

5.36.2.1 INTEGER(INTG),parameter MESH_ROUTINES::DECOMPOSITION_ALL_TYPE = 1

The decomposition contains all elements.

See also:

[MESH_ROUTINES::DecompositionTypes](#),[MESH_ROUTINES](#)

Definition at line 73 of file mesh_routines.f90.

Referenced by [MESH_ROUTINES::DECOMPOSITION_CREATE_START\(\)](#), [MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE\(\)](#), and [MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET\(\)](#).

5.36.2.2 INTEGER(INTG),parameter MESH_ROUTINES::DECOMPOSITION_CALCULATED_TYPE = 2

The element decomposition is calculated by graph partitioning.

See also:

[MESH_ROUTINES::DecompositionTypes](#),[MESH_ROUTINES](#)

Definition at line 74 of file mesh_routines.f90.

Referenced by MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE(), and MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET().

5.36.2.3 INTEGER(INTG),parameter MESH_ROUTINES::DECOMPOSITION_USER_DEFINED_TYPE = 3

The user will set the element decomposition.

See also:

[MESH_ROUTINES::DecompositionTypes](#), [MESH_ROUTINES](#)

Definition at line 75 of file mesh_routines.f90.

Referenced by MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE(), and MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET().

5.37 PROBLEM_CONSTANTS::SetupTypes

Setup type parameters.

Variables

- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM_SETUP_INITIAL_TYPE = 1**
Initial setup for a problem.
- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM_SETUP_CONTROL_TYPE = 2**
Solver setup for a problem.
- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM_SETUP SOLUTION_TYPE = 3**
Solution parameters setup for a problem.
- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM_SETUP_SOLVER_TYPE = 4**
Solver setup for a problem.

5.37.1 Detailed Description

Setup type parameters.

See also:

[PROBLEM_CONSTANTS](#)

5.37.2 Variable Documentation

5.37.2.1 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SETUP_CONTROL_TYPE = 2

Solver setup for a problem.

See also:

[PROBLEM_CONSTANTS::SetupTypes](#),[PROBLEM_ROU](#)

Definition at line 107 of file problem_constants.f90.

Referenced by `FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_PROBLEM_SETUP()`, `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP()`, `LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SETUP()`, `PROBLEM_ROUTINES::PROBLEM_CONTROL_CREATE_FINISH()`, and `PROBLEM_ROUTINES::PROBLEM_CONTROL_CREATE_START()`.

5.37.2.2 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SETUP_- INITIAL_TYPE = 1

Initial setup for a problem.

See also:

[PROBLEM_CONSTANTS::SetupTypes](#),[PROBLEM_CONSTANTS](#)

Definition at line 106 of file problem_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_PROBLEM_SETUP(), FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_PROBLEM_SUBTYPE_SET(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SUBTYPE_SET(), LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SETUP(), PROBLEM_ROUTINES::PROBLEM_CREATE_FINISH(), and PROBLEM_ROUTINES::PROBLEM_CREATE_START().

5.37.2.3 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SETUP_- SOLUTION_TYPE = 3

Solution parameters setup for a problem.

See also:

[PROBLEM_CONSTANTS::SetupTypes](#),[PROBLEM_CONSTANTS](#)

Definition at line 108 of file problem_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_PROBLEM_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(), LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SETUP(), PROBLEM_ROUTINES::PROBLEM_SOLUTION_EQUATIONS_SET_ADD(), PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_CREATE_FINISH(), and PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_CREATE_START().

5.37.2.4 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SETUP_- SOLVER_TYPE = 4

Solver setup for a problem.

See also:

[PROBLEM_CONSTANTS::SetupTypes](#),[PROBLEM_CONSTANTS](#)

Definition at line 109 of file problem_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_PROBLEM_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(), LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SETUP(), PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_FINISH(), and PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_START().

5.38 PROBLEM_CONSTANTS::SetupActionTypes

Setup action type parameters.

Variables

- INTEGER(INTG), parameter [PROBLEM_CONSTANTS::PROBLEM_SETUP_START_ACTION = 1](#)

Start setup action.

- INTEGER(INTG), parameter [PROBLEM_CONSTANTS::PROBLEM_SETUP_FINISH_ACTION = 2](#)

Finish setup action.

- INTEGER(INTG), parameter [PROBLEM_CONSTANTS::PROBLEM_SETUP_DO_ACTION = 3](#)

Do setup action.

5.38.1 Detailed Description

Setup action type parameters.

See also:

[PROBLEM_CONSTANTS](#)

5.38.2 Variable Documentation

5.38.2.1 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SETUP_DO_ACTION = 3

Do setup action.

See also:

[PROBLEM_CONSTANTS::SetupActionTypes](#),[CONSTANTS_ROUTINES](#)

Definition at line 118 of file problem_constants.f90.

Referenced by [FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_PROBLEM_SETUP\(\)](#), [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP\(\)](#), [LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SETUP\(\)](#), and [PROBLEM_ROUTINES::PROBLEM SOLUTION_EQUATIONS_SET_ADD\(\)](#).

5.38.2.2 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SETUP_FINISH_ACTION = 2

Finish setup action.

See also:

[PROBLEM_CONSTANTS::SetupActionTypes](#),[CONSTANTS_ROUTINES](#)

Definition at line 117 of file problem_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_PROBLEM_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(), LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SETUP(), PROBLEM_ROUTINES::PROBLEM_CONTROL_CREATE_FINISH(), PROBLEM_ROUTINES::PROBLEM_CREATE_FINISH(), PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_CREATE_FINISH(), and PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_FINISH().

5.38.2.3 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SETUP_START_ACTION = 1

Start setup action.

See also:

[PROBLEM_CONSTANTS::SetupActionTypes](#),[CONSTANTS_ROUTINES](#)

Definition at line 116 of file problem_constants.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_PROBLEM_SETUP(), FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_PROBLEM_SUBTYPE_SET(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SUBTYPE_SET(), LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SETUP(), PROBLEM_ROUTINES::PROBLEM_CONTROL_CREATE_START(), PROBLEM_ROUTINES::PROBLEM_CREATE_START(), PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_CREATE_START(), and PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_START().

5.39 PROBLEM_CONSTANTS::SolutionOutputTypes

Solution output types.

Variables

- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM SOLUTION NO_-
OUTPUT = 0
No output.
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM SOLUTION TIMING_-
OUTPUT = 1
Timing information output.
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM SOLUTION MATRIX_-
OUTPUT = 2
All below and solution matrices output.

5.39.1 Detailed Description

Solution output types.

See also:

[PROBLEM_CONSTANTS](#)

5.39.2 Variable Documentation

5.39.2.1 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM SOLUTION_- MATRIX_OUTPUT = 2

All below and solution matrices output.

See also:

[PROBLEM_CONSTANTS::SolutionOutputTypes](#), [PROBLEM_CONSTANTS](#)

Definition at line 127 of file problem_constants.f90.

5.39.2.2 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM SOLUTION_- NO_OUTPUT = 0

No output.

See also:

[PROBLEM_CONSTANTS::SolutionOutputTypes](#), [PROBLEM_CONSTANTS](#)

Definition at line 125 of file problem_constants.f90.

**5.39.2.3 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SOLUTION_-
TIMING_OUTPUT = 1**

Timing information output.

See also:

[PROBLEM_CONSTANTS::SolutionOutputTypes](#),[PROBLEM_CONSTANTS](#)

Definition at line 126 of file problem_constants.f90.

5.40 PROBLEM_CONSTANTS::SolutionLinearityTypes

The solution linearity type parameters.

Variables

- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM_SOLUTION_LINEAR = 1**

The problem solution is linear.

- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM_SOLUTION_NONLINEAR = 2**

The problem solution is nonlinear.

5.40.1 Detailed Description

The solution linearity type parameters.

See also:

[PROBLEM_CONSTANTS](#)

5.40.2 Variable Documentation

5.40.2.1 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SOLUTION_LINEAR = 1

The problem solution is linear.

See also:

[PROBLEM_CONSTANTS::SolutionLinearityTypes](#), [PROBLEM_CONSTANTS](#)

Definition at line 134 of file problem_constants.f90.

Referenced by [PROBLEM_ROUTINES::PROBLEM SOLUTION_INITIALISE\(\)](#), and [PROBLEM_ROUTINES::PROBLEM SOLUTION_solve\(\)](#).

5.40.2.2 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SOLUTION_NONLINEAR = 2

The problem solution is nonlinear.

See also:

[PROBLEM_CONSTANTS::SolutionLinearityTypes](#), [PROBLEM_CONSTANTS](#)

Definition at line 135 of file problem_constants.f90.

Referenced by [FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_PROBLEM_SETUP\(\)](#), [PROBLEM_ROUTINES::PROBLEM SOLUTION_JACOBIAN_EVALUATE\(\)](#),

PROBLEM_ROUTINES::PROBLEM_SOLUTION_RESIDUAL_EVALUATE(), SOLVER_-
 MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_FINISH(), and SOLVER_MATRICES_-
 ROUTINES::SOLVER_MATRICES_INITIALISE().

5.41 PROBLEM_CONSTANTS::SolverOutputTypes

Solver output types.

Variables

- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM_SOLVER_NO_OUTPUT = 0**

No output.

- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM_SOLVER_TIMING_OUTPUT = 1**

Timing information output.

- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM_SOLVER_SOLVER_OUTPUT = 2**

All below and solver output.

5.41.1 Detailed Description

Solver output types.

See also:

[PROBLEM_CONSTANTS](#)

5.41.2 Variable Documentation

5.41.2.1 INTEGER(INTG),parameter **PROBLEM_CONSTANTS::PROBLEM_SOLVER_NO_OUTPUT = 0**

No output.

See also:

[PROBLEM_CONSTANTS::SolverOutputTypes](#),[PROBLEM_CONSTANTS](#)

Definition at line 143 of file problem_constants.f90.

5.41.2.2 INTEGER(INTG),parameter **PROBLEM_CONSTANTS::PROBLEM_SOLVER_SOLVER_OUTPUT = 2**

All below and solver output.

See also:

[PROBLEM_CONSTANTS::SolverOutputTypes](#),[PROBLEM_CONSTANTS](#)

Definition at line 145 of file problem_constants.f90.

**5.41.2.3 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SOLVER_-
TIMING_OUTPUT = 1**

Timing information output.

See also:

[PROBLEM_CONSTANTS::SolverOutputTypes](#),[PROBLEM_CONSTANTS](#)

Definition at line 144 of file problem_constants.f90.

5.42 PROBLEM_CONSTANTS::SolverSparsityTypes

Solution output types.

Variables

- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_SOLVER_SPARSE_MATRICES = 1

Use sparse solver matrices.

- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_SOLVER_FULL_MATRICES = 2

Use fully populated solver matrices.

5.42.1 Detailed Description

Solution output types.

See also:

[PROBLEM_CONSTANTS](#)

5.42.2 Variable Documentation

5.42.2.1 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SOLVER_FULL_MATRICES = 2

Use fully populated solver matrices.

See also:

[PROBLEM_CONSTANTS::SolverSparsityTypes](#),[PROBLEM_CONSTANTS](#)

Definition at line 153 of file problem_constants.f90.

5.42.2.2 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SOLVER_SPARSE_MATRICES = 1

Use sparse solver matrices.

See also:

[PROBLEM_CONSTANTS::SolverSparsityTypes](#),[PROBLEM_CONSTANTS](#)

Definition at line 152 of file problem_constants.f90.

5.43 PROBLEM_ROUTINES::LinearityTypes

The linearity type parameters.

Variables

- INTEGER(INTG), parameter [PROBLEM_ROUTINES::NUMBER_OF_PROBLEM_LINEARITIES = 3](#)

The number of problem linearity types defined.

- INTEGER(INTG), parameter [PROBLEM_ROUTINES::PROBLEM_LINEAR = 1](#)

The problem is linear.

- INTEGER(INTG), parameter [PROBLEM_ROUTINES::PROBLEM_NONLINEAR = 2](#)

The problem is non-linear.

- INTEGER(INTG), parameter [PROBLEM_ROUTINES::PROBLEM_NONLINEAR_BCS = 3](#)

The problem has non-linear boundary conditions.

5.43.1 Detailed Description

The linearity type parameters.

See also:

[PROBLEM_ROUTINES](#)

5.43.2 Variable Documentation

5.43.2.1 INTEGER(INTG),parameter [PROBLEM_ROUTINES::NUMBER_OF_PROBLEM_LINEARITIES = 3](#)

The number of problem linearity types defined.

See also:

[PROBLEM_ROUTINES::LinearityTypes](#),[PROBLEM_ROUTINES](#)

Definition at line 74 of file problem_routines.f90.

5.43.2.2 INTEGER(INTG),parameter [PROBLEM_ROUTINES::PROBLEM_LINEAR = 1](#)

The problem is linear.

See also:

[PROBLEM_ROUTINES::LinearityTypes](#),[PROBLEM_ROUTINES](#)

Definition at line 75 of file problem_routines.f90.

5.43.2.3 INTEGER(INTG),parameter PROBLEM_ROUTINES::PROBLEM_NONLINEAR = 2

The problem is non-linear.

See also:

[PROBLEM_ROUTINES::LinearityTypes](#),[PROBLEM_ROUTINES](#)

Definition at line 76 of file problem_routines.f90.

**5.43.2.4 INTEGER(INTG),parameter PROBLEM_ROUTINES::PROBLEM_NONLINEAR_-
BCS = 3**

The problem has non-linear boundary conditions.

See also:

[PROBLEM_ROUTINES::LinearityTypes](#),[PROBLEM_ROUTINES](#)

Definition at line 77 of file problem_routines.f90.

5.44 PROBLEM_ROUTINES::TimeDepedenceTypes

The time dependence type parameters.

Variables

- INTEGER(INTG), parameter [PROBLEM_ROUTINES::NUMBER_OF_PROBLEM_TIME_TYPES = 3](#)

The number of problem time dependence types defined.

- INTEGER(INTG), parameter [PROBLEM_ROUTINES::PROBLEM_STATIC = 1](#)

The problem is static and has no time dependence.

- INTEGER(INTG), parameter [PROBLEM_ROUTINES::PROBLEM_DYNAMIC = 2](#)

The problem is dynamic.

- INTEGER(INTG), parameter [PROBLEM_ROUTINES::PROBLEM_QUASISTATIC = 3](#)

The problem is quasi-static.

5.44.1 Detailed Description

The time dependence type parameters.

See also:

[PROBLEM_ROUTINES](#)

5.44.2 Variable Documentation

5.44.2.1 INTEGER(INTG),parameter [PROBLEM_ROUTINES::NUMBER_OF_PROBLEM_TIME_TYPES = 3](#)

The number of problem time dependence types defined.

See also:

[PROBLEM_ROUTINES_TimeDependenceTypes](#),[PROBLEM_ROUTINES](#)

Definition at line 84 of file problem_routines.f90.

5.44.2.2 INTEGER(INTG),parameter [PROBLEM_ROUTINES::PROBLEM_DYNAMIC = 2](#)

The problem is dynamic.

See also:

[PROBLEM_ROUTINES_TimeDependenceTypes](#),[PROBLEM_ROUTINES](#)

Definition at line 86 of file problem_routines.f90.

5.44.2.3 INTEGER(INTG),parameter PROBLEM_ROUTINES::PROBLEM_QUASISTATIC = 3

The problem is quasi-static.

See also:

PROBLEM_ROUTINES_TimeDependenceTypes,[PROBLEM_ROUTINES](#)

Definition at line 87 of file problem_routines.f90.

5.44.2.4 INTEGER(INTG),parameter PROBLEM_ROUTINES::PROBLEM_STATIC = 1

The problem is static and has no time dependence.

See also:

PROBLEM_ROUTINES_TimeDependenceTypes,[PROBLEM_ROUTINES](#)

Definition at line 85 of file problem_routines.f90.

5.45 SOLVER_ROUTINES::SolverTypes

The types of a problem solver.

Variables

- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_LINEAR_TYPE = 1](#)
A linear solution solver.
- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_NONLINEAR_TYPE = 2](#)
A nonlinear solution solver.
- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_TYPE = 3](#)
A time integration solver.
- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_TYPE = 4](#)
An eigenproblem type.

5.45.1 Detailed Description

The types of a problem solver.

See also:

[SOLVER_ROUTINES](#)

5.45.2 Variable Documentation

5.45.2.1 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_TYPE = 4

A eigenproblem type.

See also:

[SOLVER_ROUTINES::SolverTypes](#),[SOLVER_ROUTINES](#)

Definition at line 75 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_CREATE_FINISH\(\)](#), [SOLVER_ROUTINES::SOLVER_INITIALISE\(\)](#), [SOLVER_ROUTINES::SOLVER_LIBRARY_SET\(\)](#), and [SOLVER_ROUTINES::SOLVER_SOLVE\(\)](#).

5.45.2.2 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_LINEAR_TYPE = 1

Linear solution solver.

See also:

[SOLVER_ROUTINES::SolverTypes](#),[SOLVER_ROUTINES](#)

Definition at line 72 of file solver_routines.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(), LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SETUP(), SOLVER_ROUTINES::SOLVER_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_INITIALISE(), SOLVER_ROUTINES::SOLVER_LIBRARY_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_TYPE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_ABSOLUTE_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_DIVERGENCE_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_MAXIMUM_ITERATIONS_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_RELATIVE_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_TYPE_SET(), and SOLVER_ROUTINES::SOLVER_SOLVE().

5.45.2.3 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_NONLINEAR_TYPE = 2

A nonlinear solution solver.

See also:

[SOLVER_ROUTINES::SolverTypes](#), [SOLVER_ROUTINES](#)

Definition at line 73 of file solver_routines.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_PROBLEM_SETUP(), SOLVER_ROUTINES::SOLVER_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_INITIALISE(), SOLVER_ROUTINES::SOLVER_LIBRARY_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_ABSOLUTE_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_ALPHA_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_MAXSTEP_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_STEPTOL_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESearch_TYPE_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_MAXIMUM_FUNCTION_EVALUATIONS_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_MAXIMUM_ITERATIONS_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_RELATIVE_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_SOLUTION_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_DELTA0_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_TYPE_SET(), and SOLVER_ROUTINES::SOLVER_SOLVE().

5.45.2.4 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_TYPE = 3

A time integration solver.

See also:

[SOLVER_ROUTINES::SolverTypes](#), [SOLVER_ROUTINES](#)

Definition at line 74 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_INITIALISE(), SOLVER_ROUTINES::SOLVER_LIBRARY_SET(), and SOLVER_ROUTINES::SOLVER_SOLVE().

5.46 SOLVER_ROUTINES::SolverLibraries

The types of solver libraries.

Variables

- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_CMISS_LIBRARY = LIBRARY_-_CMISS_TYPE**
CMISS (internal) solver library.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_PETSC_LIBRARY = LIBRARY_-_PETSC_TYPE**
PETSc solver library.

5.46.1 Detailed Description

The types of solver libraries.

See also:

[SOLVER_ROUTINES](#)

5.46.2 Variable Documentation

5.46.2.1 INTEGER(INTG),parameter **SOLVER_ROUTINES::SOLVER_CMISS_LIBRARY = LIBRARY_CMISS_TYPE**

CMISS (internal) solver library.

See also:

[SOLVER_ROUTINES::SolverLibraries](#),[SOLVER_ROUTINES](#)

Definition at line 82 of file solver_routines.f90.

Referenced by **SOLVER_ROUTINES::SOLVER_LIBRARY_SET()**, **SOLVER_ROUTINES::SOLVER_-_LINEAR_DIRECT_CREATE_FINISH()**, **SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_-_INITIALISE()**, **SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_SOLVE()**, **SOLVER_-ROUTINES::SOLVER_LINEAR_DIRECT_TYPE_SET()**, **SOLVER_ROUTINES::SOLVER_-LINEAR_ITERATIVE_CREATE_FINISH()**, **SOLVER_ROUTINES::SOLVER_LINEAR_-ITERATIVE_SOLVE()**, **SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESearch_CREATE_-FINISH()**, **SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESearch_SOLVE()**, **SOLVER_-ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_CREATE_FINISH()**, and **SOLVER_-ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_SOLVE()**.

5.46.2.2 INTEGER(INTG),parameter **SOLVER_ROUTINES::SOLVER_PETSC_LIBRARY = LIBRARY_PETSC_TYPE**

PETSc solver library.

See also:

[SOLVER_ROUTINES::SolverLibraries](#), [SOLVER_ROUTINES](#)

Definition at line 83 of file solver_routines.f90.

Referenced by FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_PROBLEM_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(), LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SETUP(), SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_INITIALISE(), SOLVER_ROUTINES::SOLVER_LIBRARY_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_SOLVE(), SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_TYPE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_INITIALISE(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_FINISH(), SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_INITIALISE(), SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_SOLVE(), and SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_INITIALISE().

5.47 SOLVER_ROUTINES::LinearSolverTypes

The types of linear solvers.

Variables

- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_SOLVE_TYPE = 1](#)

Direct linear solver type.

- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE = 2](#)

Iterative linear solver type.

5.47.1 Detailed Description

The types of linear solvers.

See also:

[SOLVER_ROUTINES](#)

5.47.2 Variable Documentation

5.47.2.1 INTEGER(INTG),parameter [SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_SOLVE_TYPE = 1](#)

Direct linear solver type.

See also:

[SOLVER_ROUTINES::LinearSolverTypes](#),[SOLVER_ROUTINES](#)

Definition at line 90 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_LIBRARY_SET\(\)](#), [SOLVER_ROUTINES::SOLVER_LINEAR_CREATE_FINISH\(\)](#), [SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_TYPE_SET\(\)](#), [SOLVER_ROUTINES::SOLVER_LINEAR_SOLVE\(\)](#), and [SOLVER_ROUTINES::SOLVER_LINEAR_TYPE_SET\(\)](#).

5.47.2.2 INTEGER(INTG),parameter [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE = 2](#)

Iterative linear solver type.

See also:

[SOLVER_ROUTINES::LinearSolverTypes](#),[SOLVER_ROUTINES](#)

Definition at line 91 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_LIBRARY_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_LINEAR_INITIALISE(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_ABSOLUTE_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_DIVERGENCE_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_MAXIMUM_ITERATIONS_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_RELATIVE_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_SOLVE(), and SOLVER_ROUTINES::SOLVER_LINEAR_TYPE_SET().

5.48 SOLVER_ROUTINES::DirectLinearSolverTypes

The types of direct linear solvers.

Variables

- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_DIRECT_LU = 1](#)
LU direct linear solver.
- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_DIRECT_CHOLESKY = 2](#)
Cholesky direct linear solver.
- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_DIRECT_SVD = 3](#)
SVD direct linear solver.

5.48.1 Detailed Description

The types of direct linear solvers.

See also:

[SOLVER_ROUTINES](#)

5.48.2 Variable Documentation

5.48.2.1 INTEGER(INTG),parameter [SOLVER_ROUTINES::SOLVER_DIRECT_CHOLESKY = 2](#)

Cholesky direct linear solver.

See also:

[SOLVER_ROUTINES::DirectLinearSolverTypes](#),[SOLVER_ROUTINES](#)

Definition at line 99 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_TYPE_SET\(\)](#).

5.48.2.2 INTEGER(INTG),parameter [SOLVER_ROUTINES::SOLVER_DIRECT_LU = 1](#)

LU direct linear solver.

See also:

[SOLVER_ROUTINES::DirectLinearSolverTypes](#),[SOLVER_ROUTINES](#)

Definition at line 98 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_INITIALISE\(\)](#), and [SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_TYPE_SET\(\)](#).

5.48.2.3 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_DIRECT_SVD = 3

SVD direct linear solver.

See also:

[SOLVER_ROUTINES::DirectLinearSolverTypes](#),[SOLVER_ROUTINES](#)

Definition at line 100 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_TYPE_SET\(\)](#).

5.49 SOLVER_ROUTINES::IterativeLinearSolverTypes

The types of iterative linear solvers.

Variables

- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_RICHARDSON = 1**
Richardson iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_CHEBYCHEV = 2**
Chebychev iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_CONJUGATE_GRADIENT = 3**
Conjugate gradient iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_BICONJUGATE_GRADIENT = 4**
Bi-conjugate gradient iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_GMRES = 5**
Generalised minimum residual iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_BICGSTAB = 6**
Stabalised bi-conjugate gradient iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_CONJGRAD_SQUARED = 7**
Conjugate gradient squared iterative solver type.

5.49.1 Detailed Description

The types of iterative linear solvers.

See also:

[SOLVER_ROUTINES](#)

5.49.2 Variable Documentation

5.49.2.1 INTEGER(INTG),parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_BICGSTAB = 6**

Stabalised bi-conjugate gradient iterative solver type.

See also:

[SOLVER_ROUTINES::IterativeLinearSolverTypes](#),[SOLVER_ROUTINES](#)

Definition at line 112 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH(), and SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET().

5.49.2.2 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_- BICONJUGATE_GRADIENT = 4

Bi-conjugate gradient iterative solver type.

See also:

[SOLVER_ROUTINES::IterativeLinearSolverTypes](#), [SOLVER_ROUTINES](#)

Definition at line 110 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH(), and SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET().

5.49.2.3 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_- CHEBYCHEV = 2

Chebychev iterative solver type.

See also:

[SOLVER_ROUTINES::IterativeLinearSolverTypes](#), [SOLVER_ROUTINES](#)

Definition at line 108 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH(), and SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET().

5.49.2.4 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_- CONJGRAD_SQUARED = 7

Conjugate gradient squared iterative solver type.

See also:

[SOLVER_ROUTINES::IterativeLinearSolverTypes](#), [SOLVER_ROUTINES](#)

Definition at line 113 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH(), and SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET().

5.49.2.5 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_- CONJUGATE_GRADIENT = 3

Conjugate gradient iterative solver type.

See also:

[SOLVER_ROUTINES::IterativeLinearSolverTypes](#), [SOLVER_ROUTINES](#)

Definition at line 109 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH(), and SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET().

5.49.2.6 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_GMRES = 5

Generalised minimum residual iterative solver type.

See also:

[SOLVER_ROUTINES::IterativeLinearSolverTypes](#), [SOLVER_ROUTINES](#)

Definition at line 111 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_INITIALISE(), and SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET().

5.49.2.7 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_RICHARDSON = 1

Richardson iterative solver type.

See also:

[SOLVER_ROUTINES::IterativeLinearSolverTypes](#), [SOLVER_ROUTINES](#)

Definition at line 107 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH(), and SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET().

5.50 SOLVER_ROUTINES::IterativePreconditionerTypes

The types of iterative preconditioners.

Variables

- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_NO_PRECONDITIONER = 0**
No preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_JACOBI_PRECONDITIONER = 1**
Jacobi preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_BLOCK_JACOBI_PRECONDITIONER = 2**
Iterative block Jacobi preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_SOR_PRECONDITIONER = 3**
Successive over relaxation preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_INCOMPLETE_CHOLESKY_PRECONDITIONER = 4**
Incomplete Cholesky preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_INCOMPLETE_LU_PRECONDITIONER = 5**
Incomplete LU preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_ADDITIVE_SCHWARZ_PRECONDITIONER = 6**
Additive Schwarz preconditioner type.

5.50.1 Detailed Description

The types of iterative preconditioners.

See also:

[SOLVER_ROUTINES](#)

5.50.2 Variable Documentation

5.50.2.1 INTEGER(INTG),parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_ADDITIVE_SCHWARZ_PRECONDITIONER = 6**

Additive Schwarz preconditioner type.

See also:

[SOLVER_ROUTINES::IterativePreconditionerTypes](#),[SOLVER_ROUTINES](#)

Definition at line 126 of file solver_routines.f90.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()`, and `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET()`.

5.50.2.2 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_BLOCK_JACOBI_PRECONDITIONER = 2

Iterative block Jacobi preconditioner type.

See also:

[SOLVER_ROUTINES::IterativePreconditionerTypes](#),[SOLVER_ROUTINES](#)

Definition at line 122 of file solver_routines.f90.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()`, and `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET()`.

5.50.2.3 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_INCOMPLETE_CHOLESKY_PRECONDITIONER = 4

Incomplete Cholesky preconditioner type.

See also:

[SOLVER_ROUTINES::IterativePreconditionerTypes](#),[SOLVER_ROUTINES](#)

Definition at line 124 of file solver_routines.f90.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()`, and `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET()`.

5.50.2.4 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_INCOMPLETE_LU_PRECONDITIONER = 5

Incomplete LU preconditioner type.

See also:

[SOLVER_ROUTINES::IterativePreconditionerTypes](#),[SOLVER_ROUTINES](#)

Definition at line 125 of file solver_routines.f90.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()`, and `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET()`.

**5.50.2.5 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_-
JACOBI_PRECONDITIONER = 1**

Jacobi preconditioner type.

See also:

[SOLVER_ROUTINES::IterativePreconditionerTypes](#), [SOLVER_ROUTINES](#)

Definition at line 121 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH\(\)](#), [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_INITIALISE\(\)](#), and [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET\(\)](#).

**5.50.2.6 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_NO_-
PRECONDITIONER = 0**

No preconditioner type.

See also:

[SOLVER_ROUTINES::IterativePreconditionerTypes](#), [SOLVER_ROUTINES](#)

Definition at line 120 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH\(\)](#), and [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET\(\)](#).

**5.50.2.7 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_SOR_-
PRECONDITIONER = 3**

Successive over relaxation preconditioner type.

See also:

[SOLVER_ROUTINES::IterativePreconditionerTypes](#), [SOLVER_ROUTINES](#)

Definition at line 123 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH\(\)](#), and [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET\(\)](#).

5.51 SOLVER_ROUTINES::NonlinearSolverTypes

The types of nonlinear solvers.

Variables

- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH = 1**

Newton line search nonlinear solver type.

- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION = 2**

Newton trust region nonlinear solver type.

5.51.1 Detailed Description

The types of nonlinear solvers.

See also:

[SOLVER_ROUTINES](#)

5.51.2 Variable Documentation

5.51.2.1 INTEGER(INTG),parameter **SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH = 1**

Newton line search nonlinear solver type.

See also:

[SOLVER_ROUTINES::NonlinearSolverTypes](#), [SOLVER_ROUTINES](#)

Definition at line 133 of file solver_routines.f90.

Referenced by **SOLVER_ROUTINES::SOLVER_LIBRARY_SET()**, **SOLVER_ROUTINES::SOLVER_NONLINEAR_CREATE_FINISH()**, **SOLVER_ROUTINES::SOLVER_NONLINEAR_INITIALISE()**, **SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_ALPHA_SET()**, **SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_MAXSTEP_SET()**, **SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_STEPTOL_SET()**, **SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_TYPE_SET()**, **SOLVER_ROUTINES::SOLVER_NONLINEAR_SOLVE()**, and **SOLVER_ROUTINES::SOLVER_NONLINEAR_TYPE_SET()**.

5.51.2.2 INTEGER(INTG),parameter **SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION = 2**

Newton trust region nonlinear solver type.

See also:

[SOLVER_ROUTINES::NonlinearSolverTypes](#), [SOLVER_ROUTINES](#)

Definition at line 134 of file solver_routines.f90.

Referenced by `SOLVER_ROUTINES::SOLVER_LIBRARY_SET()`, `SOLVER_ROUTINES::SOLVER_NONLINEAR_CREATE_FINISH()`, `SOLVER_ROUTINES::SOLVER_NONLINEAR_SOLVE()`, `SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_DELTA0_SET()`, `SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_TOLERANCE_SET()`, and `SOLVER_ROUTINES::SOLVER_NONLINEAR_TYPE_SET()`.

5.52 SOLVER_ROUTINES::NonlinearLineSearchTypes

The types line search techniques for Newton line search nonlinear solvers.

Variables

- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_NONLINEAR_NONORMS_LINESEARCH = 1**
No norms line search for Newton line search nonlinear solves.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_NONLINEAR_NO_LINESEARCH = 2**
No line search for Newton line search nonlinear solves.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_NONLINEAR_QUADRATIC_LINESEARCH = 3**
Quadratic search for Newton line search nonlinear solves.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_NONLINEAR_CUBIC_LINESEARCH = 4**
Cubic search for Newton line search nonlinear solves.

5.52.1 Detailed Description

The types line search techniques for Newton line search nonlinear solvers.

See also:

[SOLVER_ROUTINES](#)

5.52.2 Variable Documentation

5.52.2.1 INTEGER(INTG),parameter **SOLVER_ROUTINES::SOLVER_NONLINEAR_CUBIC_LINESEARCH = 4**

Cubic search for Newton line search nonlinear solves.

See also:

[SOLVER_ROUTINES::NonlinearLineSearchTypes](#),[SOLVER_ROUTINES](#)

Definition at line 144 of file solver_routines.f90.

Referenced by `SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_CREATE_FINISH()`, `SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_INITIALISE()`, and `SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_TYPE_SET()`.

**5.52.2.2 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_NONLINEAR_NO_-
LINESEARCH = 2**

No line search for Newton line search nonlinear solves.

See also:

[SOLVER_ROUTINES::NonlinearLineSearchTypes](#), [SOLVER_ROUTINES](#)

Definition at line 142 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_CREATE_FINISH\(\)](#), and [SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_TYPE_SET\(\)](#).

**5.52.2.3 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_NONLINEAR_-
NONORMS_LINESEARCH = 1**

No norms line search for Newton line search nonlinear solves.

See also:

[SOLVER_ROUTINES::NonlinearLineSearchTypes](#), [SOLVER_ROUTINES](#)

Definition at line 141 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_CREATE_FINISH\(\)](#), and [SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_TYPE_SET\(\)](#).

**5.52.2.4 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_NONLINEAR_-
QUADRATIC_LINESEARCH = 3**

Quadratic search for Newton line search nonlinear solves.

See also:

[SOLVER_ROUTINES::NonlinearLineSearchTypes](#), [SOLVER_ROUTINES](#)

Definition at line 143 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_CREATE_FINISH\(\)](#), and [SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_TYPE_SET\(\)](#).

5.53 SOLVER_ROUTINES::JacobianCalculationTypes

The Jacobian calculation types for a nonlinear solver.

Variables

- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_NONLINEAR_JACOBIAN_NOT_CALCULATED = 1**

The Jacobian values will not be calculated for the nonlinear equations set.

- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_NONLINEAR_JACOBIAN_ANALYTIC_CALCULATED = 2**

The Jacobian values will be calculated analytically for the nonlinear equations set.

- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_NONLINEAR_JACOBIAN_FD_CALCULATED = 3**

The Jacobian values will be calcualted using finite differences for the nonlinear equations set.

5.53.1 Detailed Description

The Jacobian calculation types for a nonlinear solver.

See also:

[SOLVER_ROUTINES](#)

5.53.2 Variable Documentation

5.53.2.1 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_NONLINEAR_JACOBIAN_ANALYTIC_CALCULATED = 2

The Jacobian values will be calculated analytically for the nonlinear equations set.

See also:

[SOLVER_ROUTINES::JacobianCalculationTypes](#),[SOLVER_ROUTINES](#)

Definition at line 151 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_CREATE_FINISH\(\)](#).

5.53.2.2 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_NONLINEAR_JACOBIAN_FD_CALCULATED = 3

The Jacobian values will be calcualted using finite differences for the nonlinear equations set.

See also:

[SOLVER_ROUTINES::JacobianCalculationTypes](#),[SOLVER_ROUTINES](#)

Definition at line 152 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_CREATE_FINISH(), and SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_INITIALISE().

5.53.2.3 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_NONLINEAR_- JACOBIAN_NOT_CALCULATED = 1

The Jacobian values will not be calculated for the nonlinear equations set.

See also:

[SOLVER_ROUTINES::JacobianCalculationTypes](#), [SOLVER_ROUTINES](#)

Definition at line 150 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_CREATE_FINISH().

5.54 SOLVER_ROUTINES::OutputTypes

The types of output.

Variables

- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_NO_OUTPUT = 0](#)
No output from the solver routines.
- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_TIMING_OUTPUT = 1](#)
Timing output from the solver routines.
- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_SOLVER_OUTPUT = 2](#)
Timing and solver specific output from the solver routines.
- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_MATRIX_OUTPUT = 3](#)
Timing and solver specific output and solution matrices output from the solver routines.

5.54.1 Detailed Description

The types of output.

See also:

[SOLVER_ROUTINES](#)

5.54.2 Variable Documentation

5.54.2.1 INTEGER(INTG),parameter [SOLVER_ROUTINES::SOLVER_MATRIX_OUTPUT = 3](#)

Timing and solver specific output and solution matrices output from the solver routines.

See also:

[SOLVER_ROUTINES::OutputTypes](#),[SOLVER_ROUTINES](#)

Definition at line 162 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_OUTPUT_TYPE_SET\(\)](#), and [SOLVER_ROUTINES::SOLVER_SOLVE\(\)](#).

5.54.2.2 INTEGER(INTG),parameter [SOLVER_ROUTINES::SOLVER_NO_OUTPUT = 0](#)

No output from the solver routines.

See also:

[SOLVER_ROUTINES::OutputTypes](#),[SOLVER_ROUTINES](#)

Definition at line 159 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_INITIALISE(), and SOLVER_ROUTINES::SOLVER_OUTPUT_TYPE_SET().

5.54.2.3 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_SOLVER_OUTPUT = 2

Timing and solver specific output from the solver routines.

See also:

[SOLVER_ROUTINES::OutputTypes](#),[SOLVER_ROUTINES](#)

Definition at line 161 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE(), SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_SOLVE(), and SOLVER_ROUTINES::SOLVER_OUTPUT_TYPE_SET().

5.54.2.4 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_TIMING_OUTPUT = 1

Timing output from the solver routines.

See also:

[SOLVER_ROUTINES::OutputTypes](#),[SOLVER_ROUTINES](#)

Definition at line 160 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE(), SOLVER_ROUTINES::SOLVER_OUTPUT_TYPE_SET(), and SOLVER_ROUTINES::SOLVER_SOLVE().

5.55 SOLVER_ROUTINES::SparsityTypes

The types of sparse solver matrices.

Variables

- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_SPARSE_MATRICES = 1](#)
Use sparse solver matrices.
- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_FULL_MATRICES = 2](#)
Use fully populated solver matrices.

5.55.1 Detailed Description

The types of sparse solver matrices.

See also:

[SOLVER_ROUTINES](#)

5.55.2 Variable Documentation

5.55.2.1 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_FULL_MATRICES = 2

Use fully populated solver matrices.

See also:

[SOLVER_ROUTINES::SparsityTypes](#),[SOLVER_ROUTINES](#)

Definition at line 170 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_CREATE_FINISH\(\)](#), [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH\(\)](#), [SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_CREATE_FINISH\(\)](#), and [SOLVER_ROUTINES::SOLVER_SPARSITY_TYPE_SET\(\)](#).

5.55.2.2 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_SPARSE_MATRICES = 1

Use sparse solver matrices.

See also:

[SOLVER_ROUTINES::SparsityTypes](#),[SOLVER_ROUTINES](#)

Definition at line 169 of file solver_routines.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(), LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SETUP(), SOLVER_ROUTINES::SOLVER_INITIALISE(), SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESearch_CREATE_FINISH(), and SOLVER_ROUTINES::SOLVER_SPARSITY_TYPE_SET().

5.56 TREES::TreeNodeColourTypes

The colour of the tree nodes.

Variables

- INTEGER(INTG), parameter [TREES::TREE_BLACK_NODE = 0](#)
The black colour type for a tree node.
- INTEGER(INTG), parameter [TREES::TREE_RED_NODE = 1](#)
The red colour type for a tree node.

5.56.1 Detailed Description

The colour of the tree nodes.

See also:

[TREES](#)

5.56.2 Variable Documentation

5.56.2.1 INTEGER(INTG),parameter TREES::TREE_BLACK_NODE = 0

The black colour type for a tree node.

See also:

[TREES::TreeNodeColourTypes](#),[TREES](#)

Definition at line 62 of file trees.f90.

Referenced by [TREES::TREE_ITEM_DELETE\(\)](#), [TREES::TREE_ITEM_INSERT\(\)](#), and [TREES::TREE_NODE_INITIALISE\(\)](#).

5.56.2.2 INTEGER(INTG),parameter TREES::TREE_RED_NODE = 1

The red colour type for a tree node.

See also:

[TREES::TreeNodeColourTypes](#),[TREES](#)

Definition at line 63 of file trees.f90.

Referenced by [TREES::TREE_ITEM_DELETE\(\)](#), and [TREES::TREE_ITEM_INSERT\(\)](#).

5.57 TREES::TreeNodeInsertStatus

The insert status for tree nodes.

Variables

- INTEGER(INTG), parameter [TREES::TREE_NODE_INSERT_SUCESSFUL = 1](#)
Successful insert status.
- INTEGER(INTG), parameter [TREES::TREE_NODE_DUPLICATE_KEY = 2](#)
Duplicate key found for those trees that do not allow duplicate keys.

5.57.1 Detailed Description

The insert status for tree nodes.

See also:

[TREES](#)

5.57.2 Variable Documentation

5.57.2.1 INTEGER(INTG),parameter [TREES::TREE_NODE_DUPLICATE_KEY = 2](#)

Duplicate key found for those trees that do not allow duplicate keys.

See also:

[TREES::TreeNodeInsertStatus](#), [TREES](#)

Definition at line 71 of file trees.f90.

Referenced by [TREES::TREE_ITEM_INSERT\(\)](#).

5.57.2.2 INTEGER(INTG),parameter [TREES::TREE_NODE_INSERT_SUCESSFUL = 1](#)

Successful insert status.

See also:

[TREES::TreeNodeInsertStatus](#), [TREES](#)

Definition at line 70 of file trees.f90.

Referenced by [NODE_ROUTINES::NODE_NUMBER_SET\(\)](#), and [TREES::TREE_ITEM_INSERT\(\)](#).

5.58 TREES::TreeInsertTypes

The insert type for a tree.

Variables

- INTEGER(INTG), parameter [TREES::TREE_DUPLICATES_ALLOWED_TYPE = 1](#)
Duplicate keys allowed tree type.
- INTEGER(INTG), parameter [TREES::TREE_NO_DUPLICATES_ALLOWED = 2](#)
No duplicate keys allowed tree type.

5.58.1 Detailed Description

The insert type for a tree.

See also:

[TREES](#)

5.58.2 Variable Documentation

5.58.2.1 INTEGER(INTG),parameter [TREES::TREE_DUPLICATES_ALLOWED_TYPE = 1](#)

Duplicate keys allowed tree type.

See also:

[TREES::TreeInsertTypes](#),[TREES](#)

Definition at line 78 of file trees.f90.

Referenced by [TREES::TREE_CREATE_START\(\)](#), and [TREES::TREE_INSERT_TYPE_SET\(\)](#).

5.58.2.2 INTEGER(INTG),parameter [TREES::TREE_NO_DUPLICATES_ALLOWED = 2](#)

No duplicate keys allowed tree type.

See also:

[TREES::TreeInsertTypes](#),[TREES](#)

Definition at line 79 of file trees.f90.

Referenced by [MESH_ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE\(\)](#), [NODE_ROUTINES::NODES_CREATE_START\(\)](#), [TREES::TREE_INSERT_TYPE_SET\(\)](#), and [TREES::TREE_ITEM_INSERT\(\)](#).

Chapter 6

Namespace Documentation

6.1 ANALYTIC_ANALYSIS_ROUTINES Namespace Reference

This module handles all analytic analysis routines.

Functions

- subroutine [ANALYTIC_ANALYSIS_CALCULATE](#) (FIELD, FILE_ID, ERR, ERROR,*)
Calculate the analytic analysis data.
- subroutine [ANALYTIC_ANALYSIS_EXPORT](#) (FIELD, FILE_NAME, METHOD, ERR, ERROR,*)
Export analytic information.
- subroutine [ANALYTIC_ANALYSIS_FORTRAN_FILE_OPEN](#) (FILE_ID, FILE_NAME, FILE_STATUS, ERR, ERROR,*)
Open a file using Fortran. TODO should we use method in FIELD_IO??
- subroutine [ANALYTIC_ANALYSIS_FORTRAN_FILE_WRITE_STRING](#) (FILE_ID, STRING_DATA, LEN_OF_DATA, ERR, ERROR,*)
Write a string using FORTRAN IO. TODO should we use FIELD_IO_FORTRAN_FILE_WRITE_STRING??
- subroutine [ANALYTIC_ANALYSIS_INTEGRAL_ABSOLUTE_ERROR_GET](#) (FIELD, VARIABLE_NUMBER, POWER, VALUE, ERR, ERROR,*)
Get integral absolute error value for the field.
- subroutine [ANALYTIC_ANALYSIS_INTEGRAL_ANALYTIC_VALUE_GET](#) (FIELD, VARIABLE_NUMBER, POWER, VALUE, ERR, ERROR,*)
Get integral analytic value for the field TODO should we use analytical formula to calculate the integration?
- subroutine [ANALYTIC_ANALYSIS_INTEGRAL_NUMERICAL_VALUE_GET](#) (FIELD, VARIABLE_NUMBER, POWER, VALUE, ERR, ERROR,*)
Get integral numerical value for the field, TODO check integral calculation.

- subroutine [ANALYTIC_ANALYSIS_INTEGRAL_PERCENT_ERROR_GET](#) (FIELD, VARIABLE_NUMBER, POWER, VALUE, ERR, ERROR,*)

Get integral percentage error value for the field.
- subroutine [ANALYTIC_ANALYSIS_INTEGRAL_RELATIVE_ERROR_GET](#) (FIELD, VARIABLE_NUMBER, POWER, VALUE, ERR, ERROR,*)

Get integral relative error value for the field.
- subroutine [ANALYTIC_ANALYSIS_NODE_ABSOLUTE_ERROR_GET](#) (FIELD, VARIABLE_NUMBER, COMPONENT_NUMBER, NODE_NUMBER, DERIVATIVE_NUMBER, VALUE, ERR, ERROR,*)

Get absolute error value for the node.
- subroutine [ANALYTIC_ANALYSIS_NODE_ANALYTIC_VALUE_GET](#) (FIELD, VARIABLE_NUMBER, COMPONENT_NUMBER, NODE_NUMBER, DERIVATIVE_NUMBER, VALUE, ERR, ERROR,*)

Get Analytic value for the node.
- subroutine [ANALYTIC_ANALYSIS_NODE_NUMERICAL_VALUE_GET](#) (FIELD, VARIABLE_NUMBER, COMPONENT_NUMBER, NODE_NUMBER, DERIVATIVE_NUMBER, VALUE, ERR, ERROR,*)

Get Numerical value for the node.
- subroutine [ANALYTIC_ANALYSIS_NODE_PERCENT_ERROR_GET](#) (FIELD, VARIABLE_NUMBER, COMPONENT_NUMBER, NODE_NUMBER, DERIVATIVE_NUMBER, VALUE, ERR, ERROR,*)

Get percentage error value for the node.
- subroutine [ANALYTIC_ANALYSIS_NODE_RELATIVE_ERROR_GET](#) (FIELD, VARIABLE_NUMBER, COMPONENT_NUMBER, NODE_NUMBER, DERIVATIVE_NUMBER, VALUE, ERR, ERROR,*)

Get relative error value for the node.
- subroutine [ANALYTIC_ANALYSIS_RMS_PERCENT_ERROR_GET](#) (FIELD, VARIABLE_NUMBER, VALUE, ERR, ERROR,*)

Get rms percentage error value for the field.
- subroutine [ANALYTIC_ANALYSIS_RMS_ABSOLUTE_ERROR_GET](#) (FIELD, VARIABLE_NUMBER, VALUE, ERR, ERROR,*)

Get rms absolute error value for the field.
- subroutine [ANALYTIC_ANALYSIS_RMS_RELATIVE_ERROR_GET](#) (FIELD, VARIABLE_NUMBER, VALUE, ERR, ERROR,*)

Get rms relative error value for the field.

6.1.1 Detailed Description

This module handles all analytic analysis routines.

6.1.2 Function Documentation

**6.1.2.1 subroutine ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_-
ANALYSIS_CALCULATE (TYPE(FIELD_TYPE),intent(in),pointer *FIELD*,
INTEGER(INTG),intent(in) *FILE_ID*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Calculate the analytic analysis data.

Parameters:

FILE_ID file ID
ERR The error code
ERROR The error string

Definition at line 86 of file analytic_analysis_routines.f90.

References KINDS::_DP.

**6.1.2.2 subroutine ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_EXPORT
(TYPE(FIELD_TYPE),pointer *FIELD*, TYPE(VARYING_STRING),intent(inout)
FILE_NAME, TYPE(VARYING_STRING),intent(in) *METHOD*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Export analytic information.

See also:

{[ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_EXPORT](#)}.

Parameters:

FIELD the field object
FILE_NAME file name
ERR The error code
ERROR The error string

Definition at line 188 of file analytic_analysis_routines.f90.

**6.1.2.3 subroutine ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_-
FORTRAN_FILE_OPEN (INTEGER(INTG),intent(inout) *FILE_ID*,
TYPE(VARYING_STRING),intent(inout) *FILE_NAME*, TYPE(VARYING_-
STRING),intent(in) *FILE_STATUS*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Open a file using Fortran. TODO should we use method in FIELD_IO??

Parameters:

FILE_ID file ID
FILE_NAME the name of file.
FILE_STATUS status for opening a file

ERR The error code

ERROR The error string

Definition at line 224 of file analytic_analysis_routines.f90.

**6.1.2.4 subroutine ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_-
FORTRAN_FILE_WRITE_STRING (INTEGER(INTG),intent(in)
FILE_ID, TYPE(VARYING_STRING),intent(in) STRING_DATA,
INTEGER(INTG),intent(in) LEN_OF_DATA, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Write a string using FORTRAN IO. TODO should we use FIELD_IO_FORTRAN_FILE_WRITE_-
STRING??

Parameters:

FILE_ID file ID

STRING_DATA the string data.

LEN_OF_DATA length of string

ERR The error code

ERROR The error string

Definition at line 253 of file analytic_analysis_routines.f90.

**6.1.2.5 subroutine ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_-
INTEGRAL_ABSOLUTE_ERROR_GET (TYPE(FIELD_TYPE),pointer FIELD,
INTEGER(INTG),intent(in) VARIABLE_NUMBER, INTEGER(INTG),intent(in)
POWER, REAL(DP),intent(out) VALUE, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)**

Get integral absolute error value for the field.

Parameters:

FIELD the field.

VARIABLE_NUMBER variable number

POWER power

VALUE the integral absolute error

ERR The error code

ERROR The error string

Definition at line 283 of file analytic_analysis_routines.f90.

**6.1.2.6 subroutine ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_-
INTEGRAL_ANALYTIC_VALUE_GET (TYPE(FIELD_TYPE),pointer FIELD,
INTEGER(INTG),intent(in) VARIABLE_NUMBER, INTEGER(INTG),intent(in)
POWER, REAL(DP),intent(out) VALUE, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)**

Get integral analytic value for the field TODO should we use analytical formula to calculate the integration?

Parameters:

FIELD the field.
VARIABLE_NUMBER variable number
POWER power
VALUE the integral analytic value
ERR The error code
ERROR The error string

Definition at line 317 of file analytic_analysis_routines.f90.

References KINDS::_DP.

**6.1.2.7 subroutine ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_-
 INTEGRAL_NUMERICAL_VALUE_GET (TYPE(FIELD_TYPE),pointer *FIELD*,
 INTEGER(INTG),intent(in) *VARIABLE_NUMBER*, INTEGER(INTG),intent(in)
POWER, REAL(DP),intent(out) *VALUE*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Get integral numerical value for the field, TODO check integral calculation.

Parameters:

FIELD the field.
VARIABLE_NUMBER variable number
POWER power
VALUE the integral numerical value
ERR The error code
ERROR The error string

Definition at line 367 of file analytic_analysis_routines.f90.

References KINDS::_DP.

**6.1.2.8 subroutine ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_-
 INTEGRAL_PERCENT_ERROR_GET (TYPE(FIELD_TYPE),pointer *FIELD*,
 INTEGER(INTG),intent(in) *VARIABLE_NUMBER*, INTEGER(INTG),intent(in)
POWER, REAL(DP),intent(out) *VALUE*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Get integral percentage error value for the field.

Parameters:

FIELD the field.
VARIABLE_NUMBER variable number
POWER power
VALUE the integral percentage error
ERR The error code
ERROR The error string

Definition at line 417 of file analytic_analysis_routines.f90.

References KINDS::_DP.

**6.1.2.9 subroutine ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_-
INTEGRAL_RELATIVE_ERROR_GET (TYPE(FIELD_TYPE),pointer FIELD,
INTEGER(INTG),intent(in) VARIABLE_NUMBER, INTEGER(INTG),intent(in)
POWER, REAL(DP),intent(out) VALUE, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)**

Get integral relative error value for the field.

Parameters:

FIELD the field.
VARIABLE_NUMBER variable number
POWER power
VALUE the integral relative error
ERR The error code
ERROR The error string

Definition at line 455 of file analytic_analysis_routines.f90.

References KINDS::_DP.

**6.1.2.10 subroutine ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_-
NODE_ABSOLUTE_ERROR_GET (TYPE(FIELD_TYPE),pointer FIELD,
INTEGER(INTG),intent(in) VARIABLE_NUMBER, INTEGER(INTG),intent(in)
COMPONENT_NUMBER, INTEGER(INTG),intent(in) NODE_NUMBER,
INTEGER(INTG),intent(in) DERIVATIVE_NUMBER, REAL(DP),intent(out) VALUE,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)**

Get absolute error value for the node.

Parameters:

FIELD the field.
VARIABLE_NUMBER variable number
COMPONENT_NUMBER component number
NODE_NUMBER node number
DERIVATIVE_NUMBER derivative number
VALUE the absolute error
ERR The error code
ERROR The error string

Definition at line 493 of file analytic_analysis_routines.f90.

**6.1.2.11 subroutine ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_-
NODE_ANALYTIC_VALUE_GET (TYPE(FIELD_TYPE),pointer FIELD,
INTEGER(INTG),intent(in) VARIABLE_NUMBER, INTEGER(INTG),intent(in)
COMPONENT_NUMBER, INTEGER(INTG),intent(in) NODE_NUMBER,
INTEGER(INTG),intent(in) DERIVATIVE_NUMBER, REAL(DP),intent(out) VALUE,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)**

Get Analytic value for the node.

Parameters:

FIELD the field.
VARIABLE_NUMBER variable number
COMPONENT_NUMBER component number
NODE_NUMBER node number
DERIVATIVE_NUMBER derivative number
VALUE the analytic value
ERR The error code
ERROR The error string

Definition at line 529 of file analytic_analysis_routines.f90.

References FIELD_ROUTINES::FIELD_ANALYTIC_SET_TYPE.

**6.1.2.12 subroutine ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_-
 NODE_NUMERICAL_VALUE_GET (TYPE(FIELD_TYPE),pointer *FIELD*,
 INTEGER(INTG),intent(in) *VARIABLE_NUMBER*, INTEGER(INTG),intent(in)
COMPONENT_NUMBER, INTEGER(INTG),intent(in) *NODE_NUMBER*,
 INTEGER(INTG),intent(in) *DERIVATIVE_NUMBER*, REAL(DP),intent(out) *VALUE*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Get Numerical value for the node.

Parameters:

FIELD the field.
VARIABLE_NUMBER variable number
COMPONENT_NUMBER component number
NODE_NUMBER node number
DERIVATIVE_NUMBER derivative number
VALUE the numerical value
ERR The error code
ERROR The error string

Definition at line 582 of file analytic_analysis_routines.f90.

References FIELD_ROUTINES::FIELD_VALUES_SET_TYPE.

**6.1.2.13 subroutine ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_-
 NODE_PERCENT_ERROR_GET (TYPE(FIELD_TYPE),pointer *FIELD*,
 INTEGER(INTG),intent(in) *VARIABLE_NUMBER*, INTEGER(INTG),intent(in)
COMPONENT_NUMBER, INTEGER(INTG),intent(in) *NODE_NUMBER*,
 INTEGER(INTG),intent(in) *DERIVATIVE_NUMBER*, REAL(DP),intent(out) *VALUE*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)
 [private]**

Get percentage error value for the node.

Parameters:

FIELD the field.
VARIABLE_NUMBER variable number
COMPONENT_NUMBER component number
NODE_NUMBER node number
DERIVATIVE_NUMBER derivative number
VALUE the percentage error
ERR The error code
ERROR The error string

Definition at line 637 of file analytic_analysis_routines.f90.

References KINDS::_DP.

**6.1.2.14 subroutine ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_-
 NODE_RELATIVE_ERROR_GET (TYPE(FIELD_TYPE),pointer *FIELD*,
 INTEGER(INTG),intent(in) *VARIABLE_NUMBER*, INTEGER(INTG),intent(in)
COMPONENT_NUMBER, INTEGER(INTG),intent(in) *NODE_NUMBER*,
 INTEGER(INTG),intent(in) *DERIVATIVE_NUMBER*, REAL(DP),intent(out) *VALUE*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Get relative error value for the node.

Parameters:

FIELD the field.
VARIABLE_NUMBER variable number
COMPONENT_NUMBER component number
NODE_NUMBER node number
DERIVATIVE_NUMBER derivative number
VALUE the relative error
ERR The error code
ERROR The error string

Definition at line 679 of file analytic_analysis_routines.f90.

References KINDS::_DP.

**6.1.2.15 subroutine ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_-
 RMS_ABSOLUTE_ERROR_GET (TYPE(FIELD_TYPE),pointer *FIELD*,
 INTEGER(INTG),intent(in) *VARIABLE_NUMBER*, REAL(DP),intent(out) *VALUE*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Get rms absolute error value for the field.

Parameters:

FIELD the field.
VARIABLE_NUMBER variable number

VALUE the rms absolute error

ERR The error code

ERROR The error string

Definition at line 763 of file analytic_analysis_routines.f90.

References KINDS::_DP.

**6.1.2.16 subroutine ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_-
RMS_PERCENT_ERROR_GET (TYPE(FIELD_TYPE),pointer FIELD,
INTEGER(INTG),intent(in) VARIABLE_NUMBER, REAL(DP),intent(out) VALUE,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)**

Get rms percentage error value for the field.

Parameters:

FIELD the field.

VARIABLE_NUMBER variable number

VALUE the rms percentage error

ERR The error code

ERROR The error string

Definition at line 719 of file analytic_analysis_routines.f90.

References KINDS::_DP.

**6.1.2.17 subroutine ANALYTIC_ANALYSIS_ROUTINES::ANALYTIC_ANALYSIS_-
RMS_RELATIVE_ERROR_GET (TYPE(FIELD_TYPE),pointer FIELD,
INTEGER(INTG),intent(in) VARIABLE_NUMBER, REAL(DP),intent(out) VALUE,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)**

Get rms relative error value for the field.

Parameters:

FIELD the field.

VARIABLE_NUMBER variable number

VALUE the rms relative error

ERR The error code

ERROR The error string

Definition at line 806 of file analytic_analysis_routines.f90.

References KINDS::_DP.

6.2 BASE_ROUTINES Namespace Reference

This module contains all the low-level base routines e.g., all debug, control, and low-level communication routines.

Classes

- struct [ROUTINE_LIST_ITEM_TYPE](#)
Contains information for an item in the routine list for diagnostics or timing.
- struct [ROUTINE_LIST_TYPE](#)
Contains information for the routine list for diagnostics or timing.
- struct [ROUTINE_STACK_ITEM_TYPE](#)
Contains information for an item in the routine invocation stack.
- struct [ROUTINE_STACK_TYPE](#)
Contains information for the routine invocation stack.
- interface [interface](#)
- interface [FLAG_ERROR](#)
Flags an error condition.
- interface [FLAG_WARNING](#)
Flags a warning to the user.

Functions

- subroutine [ENTERS](#) (NAME, ERR, ERROR,*)
Records the entry into the named procedure and initialises the error code.
- subroutine [ERRORS](#) (NAME, ERR, ERROR)
Records the exiting error of the subroutine.
- subroutine [EXITS](#) (NAME)
Records the exit out of the named procedure.
- subroutine [FLAG_ERROR_C](#) (STRING, ERR, ERROR,*)
Sets the error string specified by a character string and flags an error.
- subroutine [FLAG_ERROR_VS](#) (STRING, ERR, ERROR,*)
Sets the error string specified by a varying string and flags an error.
- subroutine [FLAG_WARNING_C](#) (STRING, ERR, ERROR,*)
Writes a warning message specified by a character string to the user.
- subroutine [FLAG_WARNING_VS](#) (STRING, ERR, ERROR,*)
Writes a warning message specified by a varying string to the user.

- subroutine **BASE_ROUTINES_FINALISE** (ERR, ERROR,*)

Finalises the base_routines module and deallocates all memory.
- subroutine **BASE_ROUTINES_INITIALISE** (ERR, ERROR,*)

Initialises the variables required for the base_routines module.
- subroutine **DIAGNOSTICS_SET_OFF** (ERR, ERROR,*)

Sets diagnostics off.
- subroutine **DIAGNOSTICS_SET_ON** (DIAG_TYPE, LEVEL_LIST, DIAG_FILENAME, ROUTINE_LIST, ERR, ERROR,*)

Sets diagnostics on.
- subroutine **OUTPUT_SET_OFF** (ERR, ERROR,*)

Sets writes file echo output off.
- subroutine **OUTPUT_SET_ON** (ECHO_FILENAME, ERR, ERROR,*)

Sets writes file echo output on.
- subroutine **TIMING_SET_OFF** (ERR, ERROR,*)

Sets timing off.
- subroutine **TIMING_SET_ON** (TIMING_TYPE, TIMING_SUMMARY_FLAG, TIMING_FILENAME, ROUTINE_LIST, ERR, ERROR,*)

Sets timing on.
- subroutine **TIMING_SUMMARY_OUTPUT** (ERR, ERROR,*)

Outputs the timing summary.
- subroutine **WRITE_STR** (ID, ERR, ERROR,*)

Writes the output string to a specified output stream.

Variables

- INTEGER(INTG), parameter **MAX_OUTPUT_LINES** = 500

Maximum number of lines that can be output.
- INTEGER(INTG), parameter **GENERAL_OUTPUT_TYPE** = 1

General output type.
- INTEGER(INTG), parameter **DIAGNOSTIC_OUTPUT_TYPE** = 2

Diagnostic output type.
- INTEGER(INTG), parameter **TIMING_OUTPUT_TYPE** = 3

Timing output type.
- INTEGER(INTG), parameter **ERROR_OUTPUT_TYPE** = 4

Error output type.

- INTEGER(INTG), parameter **HELP_OUTPUT_TYPE** = 5
Help output type.
- INTEGER(INTG), parameter **ECHO_FILE_UNIT** = 10
File unit for echo files.
- INTEGER(INTG), parameter **DIAGNOSTICS_FILE_UNIT** = 11
File unit for diagnostic files.
- INTEGER(INTG), parameter **TIMING_FILE_UNIT** = 12
File unit for timing files.
- INTEGER(INTG), parameter **LEARN_FILE_UNIT** = 13
File unit for learn files.
- INTEGER(INTG), parameter **IO1_FILE_UNIT** = 21
File unit for general IO 1 files.
- INTEGER(INTG), parameter **IO2_FILE_UNIT** = 22
File unit for general IO 2 files.
- INTEGER(INTG), parameter **IO3_FILE_UNIT** = 23
File unit for general IO 3 files.
- INTEGER(INTG), parameter **IO4_FILE_UNIT** = 24
File unit for general IO 4 files.
- INTEGER(INTG), parameter **IO5_FILE_UNIT** = 25
File unit for general IO 5 files.
- INTEGER(INTG), parameter **TEMPORARY_FILE_UNIT** = 80
File unit for temporary files.
- INTEGER(INTG), parameter **OPEN_COMFILE_UNIT** = 90
File unit for open command files.
- INTEGER(INTG), parameter **START_READ_COMFILE_UNIT** = 90
First file unit for read command files.
- INTEGER(INTG), parameter **STOP_READ_COMFILE_UNIT** = 99
Last file unit for read command files.
- INTEGER(INTG), parameter **ALL_DIAG_TYPE** = 1
Type for setting diagnostic output in all routines.
- INTEGER(INTG), parameter **IN_DIAG_TYPE** = 2
Type for setting diagnostic output in one routine.

- INTEGER(INTG), parameter [FROM_DIAG_TYPE](#) = 3
Type for setting diagnostic output from one routine downwards.
- INTEGER(INTG), parameter [ALL_TIMING_TYPE](#) = 1
Type for setting timing output in all routines.
- INTEGER(INTG), parameter [IN_TIMING_TYPE](#) = 2
Type for setting timing output in one routine.
- INTEGER(INTG), parameter [FROM_TIMING_TYPE](#) = 3
Type for setting timing output from one routine downwards.
- LOGICAL, save [DIAGNOSTICS](#)
.TRUE. if diagnostic output is required in any routines.
- LOGICAL, save [DIAGNOSTICS1](#)
.TRUE. if level 1 diagnostic output is active in the current routine
- LOGICAL, save [DIAGNOSTICS2](#)
.TRUE. if level 2 diagnostic output is active in the current routine
- LOGICAL, save [DIAGNOSTICS3](#)
.TRUE. if level 3 diagnostic output is active in the current routine
- LOGICAL, save [DIAGNOSTICS4](#)
.TRUE. if level 4 diagnostic output is active in the current routine
- LOGICAL, save [DIAGNOSTICS5](#)
.TRUE. if level 5 diagnostic output is active in the current routine
- LOGICAL, save [DIAGNOSTICS_LEVEL](#)
.TRUE. if the user has requested level 1 diagnostic output to be active
- LOGICAL, save [DIAGNOSTICS_LEVEL2](#)
.TRUE. if the user has requested level 2 diagnostic output to be active
- LOGICAL, save [DIAGNOSTICS_LEVEL3](#)
.TRUE. if the user has requested level 3 diagnostic output to be active
- LOGICAL, save [DIAGNOSTICS_LEVEL4](#)
.TRUE. if the user has requested level 4 diagnostic output to be active
- LOGICAL, save [DIAGNOSTICS_LEVEL5](#)
.TRUE. if the user has requested level 5 diagnostic output to be active
- LOGICAL, save [DIAG_ALL_SUBROUTINES](#)
.TRUE. if diagnostic output is required in all routines
- LOGICAL, save [DIAG_FROM_SUBROUTINE](#)
.TRUE. if diagnostic output is required from a particular routine

- LOGICAL, save **DIAG_FILE_OPEN**
.TRUE. if the diagnostic output file is open
- LOGICAL, save **DIAG_OR_TIMING**
.TRUE. if diagnostics or time is .TRUE.
- LOGICAL, save **ECHO_OUTPUT**
.TRUE. if all output is to be echoed to the echo file
- LOGICAL, save **TIMING**
.TRUE. if timing output is required in any routines.
- LOGICAL, save **TIMING_SUMMARY**
.TRUE. if timing output will be summary form via a TIMING_SUMMARY_OUTPUT call otherwise timing will be output for routines when the routine exits
- LOGICAL, save **TIMING_ALL_SUBROUTINES**
.TRUE. if timing output is required in all routines
- LOGICAL, save **TIMING_FROM_SUBROUTINE**
.TRUE. if timing output is required from a particular routine
- LOGICAL, save **TIMING_FILE_OPEN**
.TRUE. if the timing output file is open
- CHARACTER(LEN=MAXSTRLEN), save **OP_STRING**
The array of lines to output.
- TYPE(**ROUTINE_LIST_TYPE**), save **DIAG_ROUTINE_LIST**
The list of routines for which diagnostic output is required.
- TYPE(**ROUTINE_LIST_TYPE**), save **TIMING_ROUTINE_LIST**
The list of routines for which timing output is required.
- TYPE(**ROUTINE_STACK_TYPE**), save **ROUTINE_STACK**
The routine invocation stack.

6.2.1 Detailed Description

This module contains all the low-level base routines e.g., all debug, control, and low-level communication routines.

6.2.2 Function Documentation

6.2.2.1 subroutine **BASE_ROUTINES::BASE_ROUTINES_FINALISE** `(INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Finalises the base_routines module and deallocates all memory.

Todo

Finish this routine and deallocate memory.

Parameters:

ERR The error code

ERROR The error string

Definition at line 552 of file base_routines.f90.

Referenced by CMISS::CMISS_FINALISE().

Here is the caller graph for this function:

**6.2.2.2 subroutine BASE_ROUTINES::BASE_ROUTINES_INITIALISE
(INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Initialises the variables required for the base_routines module.

Parameters:

ERR The error code

ERROR The error string

Definition at line 571 of file base_routines.f90.

References KINDS::_DP, KINDS::_SP, and MACHINE_CONSTANTS::MACHINE_OS.

Referenced by CMISS::CMISS_INITIALISE().

Here is the caller graph for this function:

**6.2.2.3 subroutine BASE_ROUTINES::DIAGNOSTICS_SET_OFF
(INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Sets diagnostics off.

See also:

[BASE_ROUTINES::DIAGNOSTICS_SET_ON](#)

Parameters:

ERR The error code

ERROR The error string

Definition at line 640 of file base_routines.f90.

**6.2.2.4 subroutine BASE_ROUTINES::DIAGNOSTICS_SET_ON
(INTEGER(INTG),intent(in) *DIAG_TYPE*, INTEGER(INTG),dimension(:),intent(in)
LEVEL_LIST, CHARACTER(LEN=*),intent(in) *DIAG_FILENAME*,
CHARACTER(LEN=*),dimension(:),intent(in) *ROUTINE_LIST*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)
[private]**

Sets diagnostics on.

See also:

[BASE_ROUTINES::DIAGNOSTICS_SET_OFF](#)

Parameters:

DIAG_TYPE The type of diagnostics to set on

See also:

[BASE_ROUTINES::DiagnosticTypes](#)

LEVEL_LIST The list of diagnostic levels to set on

DIAG_FILENAME If present the name of the file to output diagnostic information to. If omitted the diagnostic output is sent to the screen

ROUTINE_LIST The list of routines to set diagnostics on in.

ERR The error code

ERROR The error string

Definition at line 695 of file base_routines.f90.

6.2.2.5 subroutine **BASE_ROUTINES::ENTERS** (CHARACTER(LEN=*),intent(in) *NAME*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Records the entry into the named procedure and initialises the error code.

See also:

[BASE_ROUTINES::EXITS](#)

Parameters:

NAME The name of the routine being entered

ERR The error code

ERROR The error string

Definition at line 218 of file base_routines.f90.

References KINDS::_SP.

Referenced by SORTING::BUBBLE_SORT_DP(), SORTING::BUBBLE_SORT_INTG(), SORTING::BUBBLE_SORT_SP(), F90C::C2FSTRING(), CLASSICAL_FIELD_ROUTINES::CL(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SET_SETUP(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELDFINITE_ELEMENT_CALCULATE(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELDFINITE_ELEMENT_JACOBIAN_EVALUATE(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELDFINITE_ELEMENT_RESIDUAL_EVALUATE(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_GET(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_SET(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_SETUP(), BINARY_FILE::CLOSE_BINARY_FILE(), BINARY_FILE::CLOSE_CMISS_BINARY_FILE(), COORDINATE_ROUTINES::CO(), COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_TYPE_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_TYPE_FINALISE().

```

TYPE_INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_-
NUMBER_GET(), COMP_ENVIRONMENT::COMPUTATIONAL_NODES_-
NUMBER_GET(), COORDINATE_ROUTINES::COORDINATE_CONVERT_-
FROM_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_-
RC_SP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_DP(), COORDINATE_-
ROUTINES::COORDINATE_CONVERT_TO_RC_SP(), COORDINATE_-
ROUTINES::COORDINATE_DELTA_CALCULATE_DP(), COORDINATE_-
ROUTINES::COORDINATE_DERIVATIVE_NORM(), COORDINATE_ROUTINES::COORDINATE_-
INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_-
INTERPOLATION_PARAMETERS_ADJUST(), COORDINATE_ROUTINES::COORDINATE_-
METRICS_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_-
CREATE_FINISH(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_-
START(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_-
NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_-
PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET_-
NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_-
SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_GET(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET_NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET_PTR(), COORDINATE_-
ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE(), COORDINATE_-
ROUTINES::COORDINATE_SYSTEM_ORIENTATION_SET_NUMBER(), COORDINATE_-
ROUTINES::COORDINATE_SYSTEM_ORIENTATION_SET_PTR(), COORDINATE_-
ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET_NUMBER(), COORDINATE_-
ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET_PTR(), COORDINATE_-
ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_-
TYPE_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_-
NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_USER_NUMBER_FIND(), COORDINATE_-
ROUTINES::COORDINATE_SYSTEMS_FINALISE(), COORDINATE_ROUTINES::COORDINATE_-
SYSTEMS_INITIALISE(), TIMER::CPU_TIMER(), MATHS::CROSS_PRODUCT_DP(), MATHS::CROSS_PRODUCT_INTG(), MATHS::CROSS_PRODUCT_SP(), COORDINATE_-
ROUTINES::D2ZX_DP(), MATHS::D_CROSS_PRODUCT_DP(), MATHS::D_CROSS_PRODUCT_-
INTG(), MATHS::D_CROSS_PRODUCT_SP(), MESH_ROUTINES::DECOMPOSITION_-
CREATE_FINISH(), MESH_ROUTINES::DECOMPOSITION_CREATE_START(), MESH_-
ROUTINES::DECOMPOSITION_DESTROY(), MESH_ROUTINES::DECOMPOSITION_-
ELEMENT_DOMAIN_CALCULATE(), MESH_ROUTINES::DECOMPOSITION_ELEMENT_-
DOMAIN_GET(), MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_-
SET(), MESH_ROUTINES::DECOMPOSITION_MESH_COMPONENT_NUMBER_-
GET(), MESH_ROUTINES::DECOMPOSITION_MESH_COMPONENT_NUMBER_SET(), MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_GET(), MESH_-
ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(), MATHS::DETERMINANT_-
FULL_DP(), MATHS::DETERMINANT_FULL_INTG(), MATHS::DETERMINANT_FULL_-
SP(), DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_ADJACENT_DOMAIN_FINALISE(), DOMAIN_-
MAPPINGS::DOMAIN_MAPPINGS_ADJACENT_DOMAIN_INITIALISE(), DOMAIN_-
MAPPINGS::DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE(), DOMAIN_-
MAPPINGS::DOMAIN_MAPPINGS_MAPPING_FINALISE(), DOMAIN_MAPPINGS::DOMAIN_-
MAPPINGS_MAPPING_GLOBAL_FINALISE(), DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_-
GLOBAL_INITIALISE(), DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_-
INITIALISE(), MESH_ROUTINES::DOMAIN_MAPPINGS_NODES_FINALISE(), MESH_-
ROUTINES::DOMAIN_MAPPINGS_NODES_INITIALISE(), MESH_ROUTINES::DOMAIN_-
TOPOLOGY_CALCULATE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_DOFS_FINALISE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_DOFS_INITIALISE(), MESH_ROUTINES::DOMAIN_-
TOPOLOGY_ELEMENT_FINALISE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_-

```

```

ELEMENT_INITIALISE(),           MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_-
FINALISE(),                     MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_INITIALISE(),
MESH_ROUTINES::DOMAIN_TOPOLOGY_FINALISE(),           MESH_ROUTINES::DOMAIN_-
TOPOLOGY_INITIALISE(),           MESH_ROUTINES::DOMAIN_TOPOLOGY_INITIALISE_-
FROM_MESH(),                    MESH_ROUTINES::DOMAIN_TOPOLOGY_LINE_FINALISE(),   MESH_-
ROUTINES::DOMAIN_TOPOLOGY_LINE_INITIALISE(),         MESH_ROUTINES::DOMAIN_-
TOPOLOGY_LINES_FINALISE(),       MESH_ROUTINES::DOMAIN_TOPOLOGY_LINES_-
INITIALISE(),                   MESH_ROUTINES::DOMAIN_TOPOLOGY_NODE_FINALISE(),   MESH_-
ROUTINES::DOMAIN_TOPOLOGY_NODE_INITIALISE(),         MESH_ROUTINES::DOMAIN_-
TOPOLOGY_NODES_FINALISE(),      MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_-
INITIALISE(),                   MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_SURROUNDING_-
ELEMENTS_CALCULATE(),          COORDINATE_ROUTINES::DXZ_DP(),                 COORDINATE_-
ROUTINES::DZX_DP(),             MATHS::EIGENVALUE_FULL_DP(),                MATHS::EIGENVALUE_-
FULL_SP(),                      MATHS::EIGENVECTOR_FULL_DP(),               MATHS::EIGENVECTOR_FULL_-
SP(),                           ELASTICITY_ROUTINES::EL(),                  ELASTICITY_ROUTINES::ELASTICITY_-
EQUATIONS_SET_SETUP(),          ELASTICITY_ROUTINES::ELASTICITYFINITE_ELEMENT_-
CALCULATE(),                   ELASTICITY_ROUTINES::ELASTICITYFINITE_ELEMENT_JACOBIAN_-
EVALUATE(),                     ELASTICITY_ROUTINES::ELASTICITYFINITE_ELEMENT_RESIDUAL_-
EVALUATE(),                     ELASTICITY_ROUTINES::ELASTICITY_PROBLEM_CLASS_TYPE_-
SET(),                          ELASTICITY_ROUTINES::ELASTICITY_PROBLEM_SETUP(),   EQUATIONS_-
MAPPING_ROUTINES::EQUATIONS_MAPPING_CALCULATE(),     EQUATIONS_MAPPING_-
ROUTINES::EQUATIONS_MAPPING_CREATE_FINISH(),        EQUATIONS_MAPPING_-
ROUTINES::EQUATIONS_MAPPING_CREATE_START(),         EQUATIONS_MAPPING_-
ROUTINES::EQUATIONS_MAPPING_CREATE_VALUES_CACHE_FINALISE(), EQUATIONS_-
MAPPING_ROUTINES::EQUATIONS_MAPPING_CREATE_VALUES_CACHE_INITIALISE(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_DESTROY(), EQUATIONS_-
MAPPING_ROUTINES::EQUATIONS_MAPPING_EQUIV_JACOBIAN_TO_VARIABLE_-
MAP_FINALISE(),                  EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_-
EQUIV_JACOBIAN_TO_VARIABLE_MAP_INITIALISE(),         EQUATIONS_MAPPING_-
ROUTINES::EQUATIONS_MAPPING_EQUIV_MATRIX_TO_VARIABLE_MAP_FINALISE(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_EQUIV_MATRIX_-
TO_VARIABLE_MAP_INITIALISE(),      EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_-
MAPPING_FINALISE(),              EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_-
INITIALISE(),                    EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_LINEAR_-
MAPPING_FINALISE(),              EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_-
LINEAR_MAPPING_INITIALISE(),     EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_-
LINEAR_MAPPING_LINEAR_MATRICES_COEFFICIENTS_SET(), EQUATIONS_MAPPING_-
ROUTINES::EQUATIONS_MAPPING_LINEAR_MATRICES_NUMBER_SET(), EQUATIONS_-
MAPPING_ROUTINES::EQUATIONS_MAPPING_LINEAR_MATRICES_VARIABLE_-
TYPES_SET(),                     EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_-
NONLINEAR_MAPPING_FINALISE(),    EQUATIONS_MAPPING_ROUTINES::EQUATIONS_-
MAPPING_NONLINEAR_MAPPING_INITIALISE(),              EQUATIONS_MAPPING_-
ROUTINES::EQUATIONS_MAPPING_RESIDUAL_COEFFICIENT_SET(), EQUATIONS_-
MAPPING_ROUTINES::EQUATIONS_MAPPING_RESIDUAL_VARIABLE_TYPE_SET(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_RHS_COEFFICIENT_SET(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_RHS_MAPPING_FINALISE(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_RHS_MAPPING_INITIALISE(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_RHS_VARIABLE_TYPE_SET(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_SOURCE_COEFFICIENT_-
SET(),                          EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_SOURCE_MAPPING_-
FINALISE(),                      EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_SOURCE_-
MAPPING_INITIALISE(),            EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_SOURCE_-
SOURCE_VARIABLE_TYPE_SET(),      EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_SOURCE_-
MAPPING_VARIABLE_TO_EQUIV_COLUMN_MAP_FINALISE(),     EQUATIONS_-

```

MAPPING_ROUTINES::EQUATIONS_MAPPING_VARIABLE_TO_EQUATIONS_JACOBIAN_MAP_FINALISE(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_VARIABLE_TO_EQUATIONS_JACOBIAN_MAP_INITIALISE(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_VARIABLE_TO_EQUATIONS_MATRICES_MAP_FINALISE(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_VARIABLE_TO_EQUATIONS_MATRICES_MAP_INITIALISE(), F90C::F2CSTRING(), FINITE_ELEMENT_ROUTINES::FEM_ELEMENT_MATRICES_FINALISE(), FINITE_ELEMENT_ROUTINES::FEM_ELEMENT_MATRICES_INITIALISE(), FIELD_ROUTINES::FI(), FIELD_IO_ROUTINES::FIE(), FIELD_ROUTINES::FIELD_COMPONENT_INTER(), FIELD_ROUTINES::FIELD_COMPONENT_INTERPOLATION_GET(), FIELD_ROUTINES::FIELD_COMPONENT_MESH_COM(), FIELD_ROUTINES::FIELD_COMPONENT_MESH_COMPONENT_GET(), FIELD_ROUTINES::FIELD_CREATE_FINISH(), FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_FINALISE(), FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_INITIALISE(), FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_GET(), FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET_NUMBER(), FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET_PTR(), FIELD_ROUTINES::FIELD_DESTROY(), FIELD_ROUTINES::FIELD_DIMENSION_GET(), FIELD_ROUTINES::FIELD_DIMENSION_SET_NUMBER(), FIELD_ROUTINES::FIELD_INTERPOLATE_GAUSS(), FIELD_ROUTINES::FIELD_INTERPOLATE_XI(), FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_FINALISE(), FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_CALCULATE(), FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_FINALISE(), FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_FINALISE(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), FIELD_IO_ROUTINES::FIELD_IO_BASIS_LHTP_FAMILY_LABEL(), FIELD_IO_ROUTINES::FIELD_IO_CREATE_FIELDS(), FIELD_IO_ROUTINES::FIELD_IO_DERIVATIVE_INFO(), FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_ATTACH_LOCAL_PROCESS(), FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_FINALIZE(), FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_INITIALISE(), FIELD_IO_ROUTINES::FIELD_IO_ELEMENTS_EXPORT(), FIELD_IO_ROUTINES::FIELD_IO_EXPORT_ELEMENTS_INTO_0(), FIELD_IO_ROUTINES::FIELD_IO_EXPORT_NODAL_GROUP_HEADER_FORTRAN(), FIELD_IO_ROUTINES::FIELD_IO_EXPORT_NODES_INTO_LOC(), FIELD_IO_ROUTINES::FIELD_IO_FIELD_INFO(), FIELD_IO_ROUTINES::FIELD_IO_FILEDS_GROUP_INFO_GET(), FIELD_IO_ROUTINES::FIELD_IO_FILEDS_IMPORT(), FIELD_IO_ROUTINES::FIELD_IO_FILL_BASIS_INFO(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_CLOSE(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_OPEN(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_DP(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_INTG(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_STRING(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_REWIND(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_DP(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_INTG(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_STRING(), FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH(), FIELD_IO_ROUTINES::FIELD_IO_LABEL_DERIVATIVE_INFO_GET(), FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET(), FIELD_IO_ROUTINES::FIELD_IO_MULTI_FILES_INFO_GET(), FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_ATTACH_LOCAL_PROCESS(), FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_FINALIZE(), FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_INITIALISE(), FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_SORT(), FIELD_IO_ROUTINES::FIELD_IO_NODES_EXPORT(), FIELD_IO_ROUTINES::FIELD_IO_TRANSLATE_LABEL_INTO_INTERPOLATION_TYPE(), FIELD_ROUTINES::FIELD_VARIABLE_COMPONENT_FINALISE(), FIELD_ROUTINES::FIELD_VARIABLE_COMPONENT_INITIALISE(), FIELD_

```

ROUTINES::FIELD_VARIABLES_FINALISE(),           FIELD_ROUTINES::FIELD_VARIABLES_-
INITIALISE(),      FIELD_ROUTINES::FIELDS_FINALISE(),      FIELD_ROUTINES::FIELDS_-
INITIALISE(),      FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_-
SET_SETUP(),       FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_-
SET_SUBTYPE_SET(), FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITYFINITE_-
ELEMENT_JACOBIAN_EVALUATE(), FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_-
FINITE_ELEMENT_RESIDUAL_EVALUATE(),          FINITE_ELASTICITY_ROUTINES::FINITE_-
ELASTICITY_GAUSS_CAUCHY_TENSOR(),            FINITE_ELASTICITY_ROUTINES::FINITE_-
ELASTICITY_GAUSS_DEFORMATION_GRADEINT_TENSOR(), FINITE_ELASTICITY_-
ROUTINES::FINITE_ELASTICITY_GAUSS_DFDZ(),     FINITE_ELASTICITY_ROUTINES::FINITE_-
ELASTICITY_GAUSS_DXDNU(),        FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_-
PROBLEM_SETUP(),          FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_-
PROBLEM_SUBTYPE_SET(),          GENERATED_MESH_ROUTINES::GENERATED_-
MESH_BASIS_GET(),             GENERATED_MESH_ROUTINES::GENERATED_MESH_-
BASIS_SET_NUMBER(),          GENERATED_MESH_ROUTINES::GENERATED_MESH_-
BASIS_SET_PTR(),             GENERATED_MESH_ROUTINES::GENERATED_MESH_CREATE_-
FINISH(),                  GENERATED_MESH_ROUTINES::GENERATED_MESH_CREATE_-
START(),                   GENERATED_MESH_ROUTINES::GENERATED_MESH_DESTROY_-
NUMBER(),                  GENERATED_MESH_ROUTINES::GENERATED_MESH_DESTROY_-
PTR(),                     GENERATED_MESH_ROUTINES::GENERATED_MESH_EXTENT_GET(), 
GENERATED_MESH_ROUTINES::GENERATED_MESH_EXTENT_SET_NUMBER(), 
GENERATED_MESH_ROUTINES::GENERATED_MESH_EXTENT_SET_PTR(), 
GENERATED_MESH_ROUTINES::GENERATED_MESH_FINALISE(),          GENERATED_-
MESH_ROUTINES::GENERATED_MESH_INITIALISE(),          GENERATED_MESH_-
ROUTINES::GENERATED_MESH_NUMBER_OF_ELEMENTS_GET(),          GENERATED_-
MESH_ROUTINES::GENERATED_MESH_NUMBER_OF_ELEMENTS_SET_NUMBER(), 
GENERATED_MESH_ROUTINES::GENERATED_MESH_NUMBER_OF_ELEMENTS_-
SET_PTR(),          GENERATED_MESH_ROUTINES::GENERATED_MESH_ORIGIN_-
GET(),              GENERATED_MESH_ROUTINES::GENERATED_MESH_ORIGIN_SET_-
NUMBER(),          GENERATED_MESH_ROUTINES::GENERATED_MESH_ORIGIN_SET_-
PTR(),              GENERATED_MESH_ROUTINES::GENERATED_MESH_REGULAR_CREATE_-
FINISH(),          GENERATED_MESH_ROUTINES::GENERATED_MESH_REGULAR_-
FINALISE(),          GENERATED_MESH_ROUTINES::GENERATED_MESH_REGULAR_-
INITIALISE(),          GENERATED_MESH_ROUTINES::GENERATED_MESH_TYPE_GET(), 
GENERATED_MESH_ROUTINES::GENERATED_MESH_TYPE_SET_NUMBER(), 
GENERATED_MESH_ROUTINES::GENERATED_MESH_TYPE_SET_PTR(),          GENERATED_-
MESH_ROUTINES::GENERATED_MESH_USER_NUMBER_FIND(),          GENERATED_-
MESH_ROUTINES::GENERATED_MESHES_FINALISE(),          GENERATED_MESH_-
ROUTINES::GENERATED_MESHES_INITIALISE(),          SORTING::HEAP_SORT_DP(), 
SORTING::HEAP_SORT_INTG(),      SORTING::HEAP_SORT_SP(),      BINARY_FILE::INQUIRE_-
EOF_BINARY_FILE(), MATHS::INVERT_FULL_DP(), MATHS::INVERT_FULL_SP(), LAPLACE_-
EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_CALCULATE(), LAPLACE_-
EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_PARAMETER_SET_UPDATE(), 
LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_SETUP(), 
LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_-
SETUP(),          LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_-
SUBTYPE_SET(),          LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATIONFINITE_-
ELEMENT_CALCULATE(),          LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-
FIXED_CONDITIONS_(),          LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-
INTE(),          LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SETUP(), 
LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(), 
LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SUBTYPE_SET(), 
LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SETUP(), 
LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SUBTYPE_-

```

SET(), LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITYFINITEELEMENTCALCULATE(), LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEMSETUP(), LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SUBTYPESET(), LISTS::LIST_CREATE_FINISH(), LISTS::LIST_CREATE_START(), LISTS::LISTDATA_TYPE_SET(), LISTS::LIST_DESTROY(), LISTS::LIST_DETACH_AND_DESTROY_DPO, LISTS::LIST_DETACH_AND_DESTROY_INTG(), LISTS::LIST_DETACH_AND_DESTROY_SP(), LISTS::LIST_FINALISE(), LISTS::LIST_INITIAL_SIZE_SET(), LISTS::LIST_INITIALISEO, LISTS::LIST_ITEM_ADD_DP1(), LISTS::LIST_ITEM_ADD_INTG1(), LISTS::LIST_ITEM_ADD_SP1(), LISTS::LIST_ITEM_DELETE(), LISTS::LIST_ITEM_IN_LIST_DP1(), LISTS::LISTITEM_IN_LIST_INTG1(), LISTS::LIST_ITEM_IN_LIST_SP1(), LISTS::LIST_NUMBER_OFITEMS_GET(), LISTS::LIST_REMOVE_DUPLICATES(), LISTS::LIST_SEARCH_DP_ARRAY(), LISTS::LIST_SEARCH_INTG_ARRAY(), LISTS::LIST_SEARCH_LINEAR_DP_ARRAY(), LISTS::LIST_SEARCH_LINEAR_INTG_ARRAY(), LISTS::LIST_SEARCH_LINEAR_SPARRAY(), LISTS::LIST_SEARCH_SP_ARRAY(), LISTS::LIST_SORT_BUBBLE_DP_ARRAY(), LISTS::LIST_SORT_BUBBLE_INTG_ARRAY(), LISTS::LIST_SORT_BUBBLE_SP_ARRAY(), LISTS::LIST_SORT_DP_ARRAY(), LISTS::LIST_SORT_HEAP_DP_ARRAY(), LISTS::LISTSORT_HEAP_INTG_ARRAY(), LISTS::LIST_SORT_HEAP_SP_ARRAY(), LISTS::LISTSORT_INTG_ARRAY(), LISTS::LIST_SORT_SHELL_DP_ARRAY(), LISTS::LIST_SORTSHELL_INTG_ARRAY(), LISTS::LIST_SORT_SHELL_SP_ARRAY(), LISTS::LIST_SORTSP_ARRAY(), STRINGS::LIST_TO_CHARACTER_C(), STRINGS::LIST_TO_CHARACTERDP(), STRINGS::LIST_TO_CHARACTER_INTG(), STRINGS::LIST_TO_CHARACTERL(), STRINGS::LIST_TO_CHARACTER_LINTG(), STRINGS::LIST_TO_CHARACTERSP(), STRINGS::LOGICAL_TO_CHARACTER(), STRINGS::LOGICAL_TO_MATRIX_ALLVALUES_SET_DP(), MATRIX_VECTOR::MATRIX_ALLVALUES_SET_INTG(), MATRIX_VECTOR::MATRIX_ALLVALUES_SET_SP(), MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_CREATE_START(), MATRIX_VECTOR::MATRIX_DATA_GETDPO, MATRIX_VECTOR::MATRIX_DATA_GET_INTG(), MATRIX_VECTOR::MATRIX_DATAGET_L(), MATRIX_VECTOR::MATRIX_DATA_GET_SP(), MATRIX_VECTOR::MATRIX_DATATYPE_SET(), MATRIX_VECTOR::MATRIX_DESTROY(), MATRIX_VECTOR::MATRIXDUPLICATE(), MATRIX_VECTOR::MATRIX_FINALISE(), MATRIX_VECTOR::MATRIXINITIALISE(), MATRIX_VECTOR::MATRIX_MAX_COLUMNS_PER_ROW_GET(), MATRIX_VECTOR::MATRIX_MAX_SIZE_SET(), MATRIX_VECTOR::MATRIX_NUMBER_NONZEROS_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATHS::MATRIX_PRODUCT_DPO, MATHS::MATRIX_PRODUCT_SP(), MATRIX_VECTOR::MATRIX_SIZE_SET(), MATRIXVECTOR::MATRIX_STORAGE_LOCATION_FIND(), MATRIX_VECTOR::MATRIX_STORAGELOCATIONS_GET(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_SET(), MATRIXVECTOR::MATRIX_STORAGE_TYPE_GET(), MATRIX_VECTOR::MATRIX_STORAGE_TYPE_SET(), MATHS::MATRIX_TRANSPOSE_DP(), MATHS::MATRIX_TRANSPOSESP(), MATRIX_VECTOR::MATRIX_VALUES_ADD_DP(), MATRIX_VECTOR::MATRIXVALUES_ADD_DP1(), MATRIX_VECTOR::MATRIX_VALUES_ADD_DP2(), MATRIXVECTOR::MATRIX_VALUES_ADD_INTG(), MATRIX_VECTOR::MATRIX_VALUES_ADDINTG1(), MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG2(), MATRIX_VECTOR::MATRIXVALUES_ADD_L(), MATRIX_VECTOR::MATRIX_VALUES_ADD_L1(), MATRIXVECTOR::MATRIX_VALUES_ADD_L2(), MATRIX_VECTOR::MATRIX_VALUES_ADDSP(), MATRIX_VECTOR::MATRIX_VALUES_ADD_SP1(), MATRIX_VECTOR::MATRIXVALUES_ADD_SP2(), MATRIX_VECTOR::MATRIX_VALUES_GET_DP(), MATRIXVECTOR::MATRIX_VALUES_GET_DP1(), MATRIX_VECTOR::MATRIX_VALUES_GETDP2(), MATRIX_VECTOR::MATRIX_VALUES_GET_INTG(), MATRIX_VECTOR::MATRIXVALUES_GET_INTG1(), MATRIX_VECTOR::MATRIX_VALUES_GET_INTG2(), MATRIXVECTOR::MATRIX_VALUES_GET_L(), MATRIX_VECTOR::MATRIX_VALUES_GET_L1(), MATRIX_VECTOR::MATRIX_VALUES_GET_L2(), MATRIX_VECTOR::MATRIX_VALUES_GET_SP(), MATRIX_VECTOR::MATRIX_VALUES_GET_SP1(), MATRIX_VECTOR::MATRIXVALUES_GET_SP2(), MATRIX_VECTOR::MATRIX_VALUES_SET_DP(), MATRIX

```

VECTOR::MATRIX_VALUES_SET_DP1(),      MATRIX_VECTOR::MATRIX_VALUES_SET_-
DP2(),    MATRIX_VECTOR::MATRIX_VALUES_SET_INTG(),   MATRIX_VECTOR::MATRIX_-
VALUES_SET_INTG1(),    MATRIX_VECTOR::MATRIX_VALUES_SET_INTG2(),   MATRIX_-
VECTOR::MATRIX_VALUES_SET_L0(),      MATRIX_VECTOR::MATRIX_VALUES_SET_L10(),
MATRIX_VECTOR::MATRIX_VALUES_SET_L20(),   MATRIX_VECTOR::MATRIX_VALUES_-
SET_SP(),   MATRIX_VECTOR::MATRIX_VALUES_SET_SP1(),   MATRIX_VECTOR::MATRIX_-
VALUES_SET_SP2(), MESH_ROUTINES::MESH_CREATE_FINISH(), MESH_ROUTINES::MESH_-
CREATE_START(),   MESH_ROUTINES::MESH_DESTROY(),   MESH_ROUTINES::MESH_-
FINALISE(),   MESH_ROUTINES::MESH_INITIALISE(),  MESH_ROUTINES::MESH_NUMBER_-
OF_COMPONENTS_GET(),      MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_-
SET_NUMBER(),      MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET_PTR(),
MESH_ROUTINES::MESH_NUMBER_OF_ELEMENTS_GET(),   MESH_ROUTINES::MESH_-
NUMBER_OF_ELEMENTS_SET_NUMBER(),   MESH_ROUTINES::MESH_NUMBER_OF_-
ELEMENTS_SET_PTR(),   MESH_ROUTINES::MESH_TOPOLOGY_CALCULATE(),   MESH_-
ROUTINES::MESH_TOPOLOGY_DOFS_CALCULATE(),      MESH_ROUTINES::MESH_-
TOPOLOGY_DOFS_FINALISE(),   MESH_ROUTINES::MESH_TOPOLOGY_DOFS_INITIALISE(),
MESH_ROUTINES::MESH_TOPOLOGY_ELEMENT_FINALISE(),   MESH_ROUTINES::MESH_-
TOPOLOGY_ELEMENT_INITIALISE(),   MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_-
ADJACENT_ELEMENTS_CALCULATE(),      MESH_ROUTINES::MESH_TOPOLOGY_-_
ELEMENTS_BASIS_SET(),   MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_-_
FINISH(),   MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_START(),   MESH_-
ROUTINES::MESH_TOPOLOGY_ELEMENTS_DESTROY(),      MESH_ROUTINES::MESH_-
TOPOLOGY_ELEMENTS_ELEMENT_BASIS_GET(),   MESH_ROUTINES::MESH_TOPOLOGY_-_
ELEMENTS_ELEMENT_BASIS_SET(),   MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_-_
ELEMENT_NODES_GET(),   MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_-_
NODES_SET(),   MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_FINALISE(),   MESH_-
ROUTINES::MESH_TOPOLOGY_ELEMENTS_INITIALISE(),   MESH_ROUTINES::MESH_-
TOPOLOGY_ELEMENTS_NUMBER_GET(),      MESH_ROUTINES::MESH_TOPOLOGY_-_
ELEMENTS_NUMBER_SET(),   MESH_ROUTINES::MESH_TOPOLOGY_FINALISE(),   MESH_-
ROUTINES::MESH_TOPOLOGY_INITIALISE(),   MESH_ROUTINES::MESH_TOPOLOGY_-_
NODE_FINALISE(),   MESH_ROUTINES::MESH_TOPOLOGY_NODE_INITIALISE(),   MESH_-
ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE(),      MESH_ROUTINES::MESH_-
TOPOLOGY_NODES_DERIVATIVES_CALCULATE(),      MESH_ROUTINES::MESH_-
TOPOLOGY_NODES_FINALISE(),      MESH_ROUTINES::MESH_TOPOLOGY_NODES_-
INITIALISE(),      MESH_ROUTINES::MESH_TOPOLOGY_NODES_SURROUNDING_-_
ELEMENTS_CALCULATE(),   MESH_ROUTINES::MESH_USER_NUMBER_FIND(),   MESH_-
ROUTINES::MESSES_FINALISE(),   MESH_ROUTINES::MESSES_INITIALISE(),   CMISS_-
MPI::MPI_ERROR_CHECK(),   NODE_ROUTINES::NODE_CHECK_EXISTS(),   NODE_-
ROUTINES::NODE_DESTROY(),   NODE_ROUTINES::NODE_INITIAL_POSITION_SET(),
NODE_ROUTINES::NODE_NUMBER_SET(),   NODE_ROUTINES::NODES_CREATE_-_
FINISH(),   NODE_ROUTINES::NODES_CREATE_START(),   NODE_ROUTINES::NODES_-_
FINALISE(),   NODE_ROUTINES::NODES_INITIALISE(),   MATHS::NORMALISE_-_
DP(),   MATHS::NORMALISE_SP(),   STRINGS::NUMBER_TO_CHARACTER_DP(),
STRINGS::NUMBER_TO_CHARACTER_INTG(),   STRINGS::NUMBER_TO_CHARACTER_-_
LINTG(), STRINGS::NUMBER_TO_CHARACTER_SP(), STRINGS::NUMBER_TO_VSTRING_SP(),
BINARY_FILE::OPEN_BINARY_FILE(), BINARY_FILE::OPEN_CMISS_BINARY_FILE(), CMISS_-
PARMETIS::PARMETIS_PARTKWAY(),   CMISS_PARMETIS::PARMETIS_PARTMESHKWAY(),
CMISS_PETSC::PETSC_ERRORHANDLING_SET_OFF(),      CMISS_PETSC::PETSC_-_
ERRORHANDLING_SET_ON(),   CMISS_PETSC::PETSC_FINALIZE(),   CMISS_PETSC::PETSC_-_
INITIALIZE(),   CMISS_PETSC::PETSC_ISCOLORINGDESTROY(),   CMISS_PETSC::PETSC_-_
ISCOLORINGFINALISE(),   CMISS_PETSC::PETSC_ISCOLORINGINITIALISE(),   CMISS_-
PETSC::PETSC_ISDESTROY(),   CMISS_PETSC::PETSC_ISFINALISE(),   CMISS_PETSC::PETSC_-_
ISINITIALISE(),   CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGAPPLY(),
CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGAPPLYIS(),      CMISS_-

```

```

PETSC::PETSC_ISLOCALTOGLOBALMAPPINGCREATE(),
ISLOCALTOGLOBALMAPPINGDESTROY(),
ISLOCALTOGLOBALMAPPINGFINALISE(),
ISLOCALTOGLOBALMAPPINGINITIALISE(),
CMISS_PETSC::PETSC_KSPDESTROY(),
CMISS_PETSC::PETSC_KSPGETCONVERGEDREASON(),
KSPGETITERATIONNUMBER(), CMISS_PETSC::PETSC_KSPGETPC(), CMISS_PETSC::PETSC_KSPCREATE(),
CMISS_PETSC::PETSC_KSPFINALISE(),
CMISS_PETSC::PETSC_KSPINITIALISE(),
CMISS_PETSC::PETSC_KSPSETFROMOPTIONS(),
KSPSETOPERATORS(), CMISS_PETSC::PETSC_KSPSETTOLERANCES(),
PETSC::PETSC_KSPSETTYPE(), CMISS_PETSC::PETSC_KSPSETUP(), CMISS_PETSC::PETSC_KSPSOLVE(),
CMISS_PETSC::PETSC_LOGPRINTSUMMARY(), CMISS_PETSC::PETSC_MATASSEMBLYBEGIN(),
CMISS_PETSC::PETSC_MATASSEMBLYEND(), CMISS_PETSC::PETSC_MATCREATE(),
CMISS_PETSC::PETSC_MATCREATETEMPIAJ(), CMISS_PETSC::PETSC_MATCREATETEMPIDENSE(),
CMISS_PETSC::PETSC_MATCREATESEQDENSE(), CMISS_PETSC::PETSC_MATDESTROY(),
CMISS_PETSC::PETSC_MATFDCOLORINGCREATE(), CMISS_PETSC::PETSC_MATFDCOLORINGDESTROY(),
CMISS_PETSC::PETSC_MATFDCOLORINGFINALISE(),
CMISS_PETSC::PETSC_MATFDCOLORINGINITIALISE(), CMISS_PETSC::PETSC_MATFFINALISE(),
MATFDCOLORINGSETFROMOPTIONS(), CMISS_PETSC::PETSC_MATGETARRAY(),
CMISS_PETSC::PETSC_MATGETCOLORING(), CMISS_PETSC::PETSC_MATGETTOWNSHIPRANGE(),
CMISS_PETSC::PETSC_MATINITIALISE(), CMISS_PETSC::PETSC_MATRESTOREARRAY(),
CMISS_PETSC::PETSC_MATSETLOCALTOGLOBALMAPPING(),
CMISS_PETSC::PETSC_MATSETOPTION(), CMISS_PETSC::PETSC_MATSETSIZES(),
CMISS_PETSC::PETSC_MATSETVALUE(), CMISS_PETSC::PETSC_MATSETVALUELOCAL(),
CMISS_PETSC::PETSC_MATSETVALUES(), CMISS_PETSC::PETSC_MATSETVALUESLOCAL(),
CMISS_PETSC::PETSC_MATVIEW(), CMISS_PETSC::PETSC_MATZEROENTRIES(),
CMISS_PETSC::PETSC_PCFINALISE(), CMISS_PETSC::PETSC_PCINITIALISE(),
CMISS_PETSC::PETSC_PCSETTYPE(), CMISS_PETSC::PETSC_SNESCREATE(),
CMISS_PETSC::PETSC_SNESDESTROY(), CMISS_PETSC::PETSC_SNESFINALISE(),
CMISS_PETSC::PETSC_SNESGETCONVERGEDREASON(), CMISS_PETSC::PETSC_SNESGETFUNCTIONNORM(),
CMISS_PETSC::PETSC_SNESGETITERATIONNUMBER(),
CMISS_PETSC::PETSC_SNESINITIALISE(), CMISS_PETSC::PETSC_SNESLINESEARCHSET(),
CMISS_PETSC::PETSC_SNESLINESEARCHSETPARAMS(), CMISS_PETSC::PETSC_SNESSETFROMOPTIONS(),
CMISS_PETSC::PETSC_SNESSETFUNCTION(), CMISS_PETSC::PETSC_SNESSETJACOBIAN(),
CMISS_PETSC::PETSC_SNESSETJACOBIAN_MATFDCOLORING(), CMISS_PETSC::PETSC_SNESSETJACOBIAN_SOLVER(),
CMISS_PETSC::PETSC_SNESSETTOLERANCES(), CMISS_PETSC::PETSC_SNESSETTRUSTREGIONTOLERANCE(),
CMISS_PETSC::PETSC_SNESSETTYPE(), CMISS_PETSC::PETSC_SNESSOLVE(), CMISS_PETSC::PETSC_VECASSEMBLYBEGIN(),
CMISS_PETSC::PETSC_VECASSEMBLYEND(), CMISS_PETSC::PETSC_VCCREATE(),
CMISS_PETSC::PETSC_VCCREATEGHOST(), CMISS_PETSC::PETSC_VCCREATEGHOSTWITHARRAY(),
CMISS_PETSC::PETSC_VCCREATEMPI(), CMISS_PETSC::PETSC_VCCREATEMPIWITHARRAY(),
CMISS_PETSC::PETSC_VCCREATESEQ(), CMISS_PETSC::PETSC_VCCREATESEQWITHARRAY(),
CMISS_PETSC::PETSC_VECDESTROY(), CMISS_PETSC::PETSC_VCDUPLICATE(),
CMISS_PETSC::PETSC_VECFINALISE(), CMISS_PETSC::PETSC_VECGETARRAY(),
PETSC::PETSC_VECGETARRAYF90(), CMISS_PETSC::PETSC_VECGETLOCALSIZE(),
CMISS_PETSC::PETSC_VECGETTOWNSHIPRANGE(), CMISS_PETSC::PETSC_VECGETSIZE(),
CMISS_PETSC::PETSC_VECGETVALUES(), CMISS_PETSC::PETSC_VECGHOSTGETLOCALFORM(),
CMISS_PETSC::PETSC_VECGHOSTUPDATEBEGIN(), CMISS_PETSC::PETSC_VECGHOSTUPDATEEND(),
CMISS_PETSC::PETSC_VECINITIALISE(), CMISS_PETSC::PETSC_VECRESTOREARRAY(),
CMISS_PETSC::PETSC_VECRESTOREARRAYF90(), CMISS_PETSC::PETSC_VECRESTOREARRAYF90()

```

```
PETSC::PETSC_VECSET(),      CMISS_PETSC::PETSC_VECSETFROMOPTIONS(),      CMISS_-
PETSC::PETSC_VECSETLOCALTOGLOBALMAPPING(),      CMISS_PETSC::PETSC_-
VECSETSIZES(),      CMISS_PETSC::PETSC_VECSETVALUES(),      CMISS_PETSC::PETSC_-
VECSETVALUESLOCAL(),      CMISS_PETSC::PETSC_VECVIEW(),      PROBLEM_-
ROUTINES::PROBLEM_CONTROL_CREATE_FINISH(),      PROBLEM_ROUTINES::PROBLEM_-
CONTROL_CREATE_START(),      PROBLEM_ROUTINES::PROBLEM_CONTROL_-
DESTROY(),      PROBLEM_ROUTINES::PROBLEM_CONTROL_FINALISE(),      PROBLEM_-
ROUTINES::PROBLEM_CONTROL_INITIALISE(),      PROBLEM_ROUTINES::PROBLEM_-
CREATE_FINISH(),      PROBLEM_ROUTINES::PROBLEM_CREATE_START(),      PROBLEM_-
ROUTINES::PROBLEM_DESTROY_NUMBER(),      PROBLEM_ROUTINES::PROBLEM_-
DESTROY_PTR(),      PROBLEM_ROUTINES::PROBLEM_FINALISE(),      PROBLEM_-
ROUTINES::PROBLEM_INITIALISE(),      PROBLEM_ROUTINES::PROBLEM_SETUP(), 
PROBLEM_ROUTINES::PROBLEM_SOLUTION_EQUATIONS_SET_ADD(),      PROBLEM_-
ROUTINES::PROBLEM_SOLUTION_FINALISE(),      PROBLEM_ROUTINES::PROBLEM_-
SOLUTION_INITIALISE(),      PROBLEM_ROUTINES::PROBLEM_SOLUTION_JACOBIAN_-
EVALUATE(),      PROBLEM_ROUTINES::PROBLEM_SOLUTION_RESIDUAL_EVALUATE(), 
PROBLEM_ROUTINES::PROBLEM_SOLUTION_SOLVE(),      PROBLEM_ROUTINES::PROBLEM_-
SOLUTIONS_CREATE_FINISH(),      PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_CREATE_-_
START(),      PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_FINALISE(),      PROBLEM_-
ROUTINES::PROBLEM_SOLUTIONS_INITIALISE(),      PROBLEM_ROUTINES::PROBLEM_-
SOLVE(),      PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_FINISH(),      PROBLEM_-
ROUTINES::PROBLEM_SOLVER_CREATE_START(),      PROBLEM_ROUTINES::PROBLEM_-
SOLVER_DESTROY(),      PROBLEM_ROUTINES::PROBLEM_SOLVER_GET(),      PROBLEM_-
ROUTINES::PROBLEM_SPECIFICATION_GET_NUMBER(), PROBLEM_ROUTINES::PROBLEM_-
SPECIFICATION_GET_PTR(),      PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_-_
NUMBER(),      PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_PTR(),      PROBLEM_-
ROUTINES::PROBLEM_USER_NUMBER_FIND(),      PROBLEM_ROUTINES::PROBLEMS_-_
FINALISE(),      PROBLEM_ROUTINES::PROBLEMS_INITIALISE(),      BINARY_FILE::READ_-_
BINARY_FILE_CHARACTER(),      BINARY_FILE::READ_BINARY_FILE_DP(),      BINARY_-_
FILE::READ_BINARY_FILE_DP1(),      BINARY_FILE::READ_BINARY_FILE_DPC(),      BINARY_-_
FILE::READ_BINARY_FILE_DPC1(),      BINARY_FILE::READ_BINARY_FILE_INTG(),      BINARY_-_
FILE::READ_BINARY_FILE_INTG1(), BINARY_FILE::READ_BINARY_FILE_LINTG(), BINARY_-_
FILE::READ_BINARY_FILE_LINTG1(),      BINARY_FILE::READ_BINARY_FILE_LOGICAL(), 
BINARY_FILE::READ_BINARY_FILE_LOGICAL1(),      BINARY_FILE::READ_BINARY_-_
FILE_SINTG(),      BINARY_FILE::READ_BINARY_FILE_SINTG1(),      BINARY_FILE::READ_-_
BINARY_FILE_SP(),      BINARY_FILE::READ_BINARY_FILE_SP1(),      BINARY_FILE::READ_-_
BINARY_FILE_SPC(),      BINARY_FILE::READ_BINARY_FILE_SPC1(),      BINARY_FILE::READ_-_
BINARY_TAG_HEADER(),      REGION_ROUTINES::REGION_COORDINATE_SYSTEM_GET(), 
REGION_ROUTINES::REGION_COORDINATE_SYSTEM_SET_NUMBER(),      REGION_-_
ROUTINES::REGION_COORDINATE_SYSTEM_SET_PTR(),      REGION_ROUTINES::REGION_-_
CREATE_FINISH(),      REGION_ROUTINES::REGION_CREATE_START(),      REGION_-_
ROUTINES::REGION_DESTROY(),      REGION_ROUTINES::REGION_LABEL_GET(),      REGION_-_
ROUTINES::REGION_LABEL_SET_NUMBER(),      REGION_ROUTINES::REGION_LABEL_-_
SET_PTR(),      REGION_ROUTINES::REGION_SUB_REGION_CREATE_FINISH(),      REGION_-_
ROUTINES::REGION_SUB_REGION_CREATE_START(),      REGION_ROUTINES::REGION_-_
USER_NUMBER_FIND(),      REGION_ROUTINES::REGION_USER_NUMBER_FIND_PTR(), 
REGION_ROUTINES::REGIONS_FINALISE(),      REGION_ROUTINES::REGIONS_INITIALISE(), 
BINARY_FILE::RESET_BINARY_NUMBER_TAGS(),      BINARY_FILE::SET_BINARY_FILE(), 
SORTING::SHELL_SORT_DP(), SORTING::SHELL_SORT_INTG(), SORTING::SHELL_SORT_SP(), 
BINARY_FILE::SKIP_BINARY_FILE(),      BINARY_FILE::SKIP_BINARY_TAGS(),      BINARY_-_
FILE::SKIP_CM_BINARY_HEADER(),      MATHS::SOLVE_SMALL_LINEAR_SYSTEM_DP(), 
MATHS::SOLVE_SMALL_LINEAR_SYSTEM_SP(),      SOLVER_ROUTINES::SOLVER_CREATE_-_
FINISH(),      SOLVER_ROUTINES::SOLVER_CREATE_START(),      SOLVER_ROUTINES::SOLVER_-_
DESTROY(),      SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_CREATE_FINISH(),      SOLVER_-
```

```

ROUTINES::SOLVER_EIGENPROBLEM_FINALISE(),           SOLVER_ROUTINES::SOLVER_-
EIGENPROBLEM_INITIALISE(),           SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_-
SOLVE(),      SOLVER_ROUTINES::SOLVER_FINALISE(),      SOLVER_ROUTINES::SOLVER_-
INITIALISE(), SOLVER_ROUTINES::SOLVER_LIBRARY_SET(), SOLVER_ROUTINES::SOLVER_-
LINEAR_CREATE_FINISH(),      SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_CREATE_-
FINISH(),      SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_FINALISE(),      SOLVER_-
ROUTINES::SOLVER_LINEAR_DIRECT_INITIALISE(),      SOLVER_ROUTINES::SOLVER_-
LINEAR_DIRECT_SOLVE(),      SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_TYPE_SET(), 
SOLVER_ROUTINES::SOLVER_LINEAR_FINALISE(),      SOLVER_ROUTINES::SOLVER_-
LINEAR_INITIALISE(),      SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_ABSOLUTE_-
TOLERANCE_SET(),      SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_-
FINISH(),      SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_DIVERGENCE_TOLERANCE_-
SET(),      SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_FINALISE(),      SOLVER_-
ROUTINES::SOLVER_LINEAR_ITERATIVE_INITIALISE(),      SOLVER_ROUTINES::SOLVER_-
LINEAR_ITERATIVE_MAXIMUM_ITERATIONS_SET(),      SOLVER_ROUTINES::SOLVER_-
LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET(),      SOLVER_ROUTINES::SOLVER_-
LINEAR_ITERATIVE_RELATIVE_TOLERANCE_SET(),      SOLVER_ROUTINES::SOLVER_-
LINEAR_ITERATIVE_SOLVE(),      SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_-
SET(),      SOLVER_ROUTINES::SOLVER_LINEAR_SOLVE(),      SOLVER_ROUTINES::SOLVER_-
LINEAR_TYPE_SET(),      SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE(), 
SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_FINISH(),      SOLVER_-
MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_START(),      SOLVER_MATRICES_-
ROUTINES::SOLVER_MATRICES_DESTROY(),      SOLVER_MATRICES_ROUTINES::SOLVER_-
MATRICES_FINALISE(),      SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_INITIALISE(), 
SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_LIBRARY_TYPE_GET(),      SOLVER_-
MATRICES_ROUTINES::SOLVER_MATRICES_LIBRARY_TYPE_SET(),      SOLVER_MATRICES_-
ROUTINES::SOLVER_MATRICES_OUTPUT(),      SOLVER_MATRICES_ROUTINES::SOLVER_-
MATRICES_STORAGE_TYPE_GET(),      SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_-
STORAGE_TYPE_SET(),      SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_FINALISE(), 
SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_FORM(),      SOLVER_MATRICES_-
ROUTINES::SOLVER_MATRIX_INITIALISE(),      SOLVER_MATRICES_ROUTINES::SOLVER_-
MATRIX_STRUCTURE_CALCULATE(),      SOLVER_ROUTINES::SOLVER_NONLINEAR_-
ABSOLUTE_TOLERANCE_SET(),      SOLVER_ROUTINES::SOLVER_NONLINEAR_-
CREATE_FINISH(),      SOLVER_ROUTINES::SOLVER_NONLINEAR_FINALISE(),      SOLVER_-
ROUTINES::SOLVER_NONLINEAR_INITIALISE(),      SOLVER_ROUTINES::SOLVER_-
NONLINEAR_JACOBIAN_EVALUATE(),      SOLVER_ROUTINES::SOLVER_NONLINEAR_-
LINESEARCH_ALPHA_SET(),      SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_-
CREATE_FINISH(),      SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_-
FINALISE(),      SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_INITIALISE(), 
SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_MAXSTEP_SET(),      SOLVER_-
ROUTINES::SOLVER_NONLINEAR_LINESEARCH_SOLVE(),      SOLVER_ROUTINES::SOLVER_-
NONLINEAR_LINESEARCH_STEPTOL_SET(),      SOLVER_ROUTINES::SOLVER_NONLINEAR_-
LINESEARCH_TYPE_SET(),      SOLVER_ROUTINES::SOLVER_NONLINEAR_MAXIMUM_-
FUNCTION_EVALUATIONS_SET(),      SOLVER_ROUTINES::SOLVER_NONLINEAR_-
MAXIMUM_ITERATIONS_SET(),      SOLVER_ROUTINES::SOLVER_NONLINEAR_-
RELATIVE_TOLERANCE_SET(),      SOLVER_ROUTINES::SOLVER_NONLINEAR_RESIDUAL_-
EVALUATE(),      SOLVER_ROUTINES::SOLVER_NONLINEAR SOLUTION_TOLERANCE_SET(), 
SOLVER_ROUTINES::SOLVER_NONLINEAR_SOLVE(),      SOLVER_ROUTINES::SOLVER_-
NONLINEAR_TRUSTREGION_CREATE_FINISH(),      SOLVER_ROUTINES::SOLVER_-
NONLINEAR_TRUSTREGION_DELTA0_SET(),      SOLVER_ROUTINES::SOLVER_NONLINEAR_-
TRUSTREGION_FINALISE(),      SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_-
INITIALISE(),      SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_SOLVE(), 
SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_TOLERANCE_SET(),      SOLVER_-
ROUTINES::SOLVER_NONLINEAR_TYPE_SET(),      SOLVER_ROUTINES::SOLVER_OUTPUT_-

```

TYPE_SET(), SOLVER_ROUTINES::SOLVER_SOLVE(), SOLVER_ROUTINES::SOLVER_SPARSITY_TYPE_SET(), SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_FINALISE(), SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_INITIALISE(), SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_solve(), SOLVER_ROUTINES::SOLVER_VARIABLES_UPDATE(), STRINGS::STRING_TO_DOUBLE_C(), STRINGS::STRING_TO_DOUBLE_VS(), STRINGS::STRING_TO_INTEGER_C(), STRINGS::STRING_TO_INTEGER_VS(), STRINGS::STRING_TO_LOGICAL_C(), STRINGS::STRING_TO_LOGICAL_VS(), STRINGS::STRING_TO_LONG_INTEGER_C(), STRINGS::STRING_TO_LONG_INTEGER_VS(), FIELD_IO_ROUTINES::STRING_TO_MUTI_INTEGERS_VS(), FIELD_IO_ROUTINES::STRING_TO_MUTI_REALS_VS(), STRINGS::STRING_TO_SINGLE_C(), STRINGS::STRING_TO_SINGLE_VS(), TREES::TREE_CREATE_FINISH(), TREES::TREE_CREATE_START(), TREES::TREE_DESTROY(), TREES::TREE_DETACH_AND_DESTROY(), TREES::TREE_DETACH_IN_ORDER(), TREES::TREE_FINALISE(), TREES::TREE_INITIALISE(), TREES::TREE_INSERT_TYPE_SET(), TREES::TREE_ITEM_DELETE(), TREES::TREE_ITEM_INSERT(), TREES::TREE_NODE_FINALISE(), TREES::TREE_NODE_INITIALISE(), TREES::TREE_NODE_KEY_GET(), TREES::TREE_NODE_VALUE_GET(), TREES::TREE_NODE_VALUE_SET(), TREES::TREE_OUTPUT(), TREES::TREE_OUTPUT_IN_ORDER(), TREES::TREE_PREDECESSOR(), TREES::TREE_SEARCH(), TREES::TREE_SUCCESSOR(), MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_DP(), MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_INTG(), MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_SP(), MATRIX_VECTOR::VECTOR_CREATE_FINISH(), MATRIX_VECTOR::VECTOR_CREATE_START(), MATRIX_VECTOR::VECTOR_DATA_GET_DP(), MATRIX_VECTOR::VECTOR_DATA_GET_INTG(), MATRIX_VECTOR::VECTOR_DATA_GET_L(), MATRIX_VECTOR::VECTOR_DATA_GET_SP(), MATRIX_VECTOR::VECTOR_DATA_TYPE_SET(), MATRIX_VECTOR::VECTOR_DESTROY(), MATRIX_VECTOR::VECTOR_DUPLICATE(), MATRIX_VECTOR::VECTOR_FINALISE(), MATRIX_VECTOR::VECTOR_INITIALISE(), MATRIX_VECTOR::VECTOR_SIZE_SET(), MATRIX_VECTOR::VECTOR_VALUES_GET_DP(), MATRIX_VECTOR::VECTOR_VALUES_GET_DP1(), MATRIX_VECTOR::VECTOR_VALUES_GET_INTG(), MATRIX_VECTOR::VECTOR_VALUES_GET_L(), MATRIX_VECTOR::VECTOR_VALUES_GET_L1(), MATRIX_VECTOR::VECTOR_VALUES_GET_SP(), MATRIX_VECTOR::VECTOR_VALUES_GET_SP1(), MATRIX_VECTOR::VECTOR_VALUES_SET_DP(), MATRIX_VECTOR::VECTOR_VALUES_SET_DP1(), MATRIX_VECTOR::VECTOR_VALUES_SET_INTG(), MATRIX_VECTOR::VECTOR_VALUES_SET_SP(), MATRIX_VECTOR::VECTOR_VALUES_SET_SP1(), BINARY_FILE::WRITE_BINARY_FILE_CHARACTER(), BINARY_FILE::WRITE_BINARY_FILE_DP(), BINARY_FILE::WRITE_BINARY_FILE_DP1(), BINARY_FILE::WRITE_BINARY_FILE_DPC(), BINARY_FILE::WRITE_BINARY_FILE_DPC1(), BINARY_FILE::WRITE_BINARY_FILE_INTG(), BINARY_FILE::WRITE_BINARY_FILE_INTG1(), BINARY_FILE::WRITE_BINARY_FILE_LINTG(), BINARY_FILE::WRITE_BINARY_FILE_LINTG1(), BINARY_FILE::WRITE_BINARY_FILE_LOGICAL(), BINARY_FILE::WRITE_BINARY_FILE_LOGICAL1(), BINARY_FILE::WRITE_BINARY_FILE_SINTG(), BINARY_FILE::WRITE_BINARY_FILE_SINTG1(), BINARY_FILE::WRITE_BINARY_FILE_SP(), BINARY_FILE::WRITE_BINARY_FILE_SP1(), BINARY_FILE::WRITE_BINARY_FILE_SPC(), BINARY_FILE::WRITE_BINARY_FILE_SPC1(), and BINARY_FILE::WRITE_BINARY_TAG_HEADER().

6.2.2.6 subroutine BASE_ROUTINES::ERRORS (CHARACTER(LEN=*),intent(in) NAME, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(inout) ERROR)

Records the exiting error of the subroutine.

Parameters:

NAME The name of the routine with an error condition
ERR The error code
ERROR The error string

Definition at line 334 of file base_routines.f90.

References MACHINE_CONSTANTS::ERROR_SEPARATOR_CONSTANT.

Referenced by SORTING::BUBBLE_SORT_DP(), SORTING::BUBBLE_SORT_INTG(), SORTING::BUBBLE_SORT_SP(), F90C::C2FSTRING(), CLASSICAL_FIELD_ROUTINES::CL(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SET_SETUP(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELDFINITE_ELEMENT_CALCULATE(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELDFINITE_ELEMENT_JACOBIAN_EVALUATE(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELDFINITE_ELEMENT_RESIDUAL_EVALUATE(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_GET(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_SET(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_SETUP(), BINARY_FILE::CLOSE_BINARY_FILE(), BINARY_FILE::CLOSE_CMISS_BINARY_FILE(), COORDINATE_ROUTINES::CO(), COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_TYPE_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_TYPE_INITIALISE(), NUMBER_GET(), NUMBER_GET(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_SP(), COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE_DP(), COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_NORM(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_PARAMETERS_ADJUST(), COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_FINISH(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_START(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET_NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_GET(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET_NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIENTATION_SET_NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIENTATION_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET_NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_USER_NUMBER_FIND(), COORDINATE_

```

ROUTINES::COORDINATE_SYSTEMS_FINALISE(), COORDINATE_ROUTINES::COORDINATE_
SYSTEMS_INITIALISE(), TIMER::CPU_TIMER(), MATHS::CROSS_PRODUCT_DP(),
MATHS::CROSS_PRODUCT_INTG(), MATHS::CROSS_PRODUCT_SP(), COORDINATE_
ROUTINES::D2ZX_DP(), MATHS::D_CROSS_PRODUCT_DP(), MATHS::D_CROSS_PRODUCT_
INTG(), MATHS::D_CROSS_PRODUCT_SP(), MESH_ROUTINES::DECOMPOSITION_
CREATE_FINISH(), MESH_ROUTINES::DECOMPOSITION_CREATE_START(), MESH_
ROUTINES::DECOMPOSITION_DESTROY(), MESH_ROUTINES::DECOMPOSITION_
ELEMENT_DOMAIN_CALCULATE(), MESH_ROUTINES::DECOMPOSITION_ELEMENT_
DOMAIN_GET(), MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_
SET(), MESH_ROUTINES::DECOMPOSITION_MESH_COMPONENT_NUMBER_
GET(), MESH_ROUTINES::DECOMPOSITION_MESH_COMPONENT_NUMBER_SET(),
MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_GET(), MESH_
ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(), MATHS::DETERMINANT_
FULL_DP(), MATHS::DETERMINANT_FULL_INTG(), MATHS::DETERMINANT_FULL_
SP(), DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_ADJACENT_DOMAIN_FINALISE(),
DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_ADJACENT_DOMAIN_INITIALISE(),
DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE(),
DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_FINALISE(),
DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_GLOBAL_FINALISE(),
DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_GLOBAL_INITIALISE(),
DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_NODES_FINALISE(), MESH_
ROUTINES::DOMAIN_MAPPINGS_NODES_INITIALISE(), MESH_ROUTINES::DOMAIN_
TOPOLOGY_CALCULATE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_DOFS_FINALISE(),
MESH_ROUTINES::DOMAIN_TOPOLOGY_DOFS_INITIALISE(), MESH_ROUTINES::DOMAIN_
TOPOLOGY_ELEMENT_FINALISE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_
FINALISE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_INITIALISE(),
MESH_ROUTINES::DOMAIN_TOPOLOGY_FINALISE(), MESH_ROUTINES::DOMAIN_
TOPOLOGY_INITIALISE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_INITIALISE_
FROM_MESH(), MESH_ROUTINES::DOMAIN_TOPOLOGY_LINE_FINALISE(), MESH_
ROUTINES::DOMAIN_TOPOLOGY_LINE_INITIALISE(), MESH_ROUTINES::DOMAIN_
TOPOLOGY_LINES_FINALISE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_LINES_
INITIALISE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_NODE_FINALISE(), MESH_
ROUTINES::DOMAIN_TOPOLOGY_NODE_INITIALISE(), MESH_ROUTINES::DOMAIN_
TOPOLOGY_NODES_FINALISE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_
INITIALISE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_SURROUNDING_
ELEMENTS_CALCULATE(), COORDINATE_ROUTINES::DXZ_DP(), COORDINATE_
ROUTINES::DZX_DP(), MATHS::EIGENVALUE_FULL_DP(), MATHS::EIGENVALUE_
FULL_SP(), MATHS::EIGENVECTOR_FULL_DP(), MATHS::EIGENVECTOR_FULL_
SP(), ELASTICITY_ROUTINES::EL(), ELASTICITY_ROUTINES::ELASTICITY_
EQUATIONS_SET_SETUP(), ELASTICITY_ROUTINES::ELASTICITYFINITE_ELEMENT_
CALCULATE(), ELASTICITY_ROUTINES::ELASTICITYFINITE_ELEMENT_JACOBIAN_
EVALUATE(), ELASTICITY_ROUTINES::ELASTICITYFINITE_ELEMENT_RESIDUAL_
EVALUATE(), ELASTICITY_ROUTINES::ELASTICITY_PROBLEM_CLASS_TYPE_
SET(), ELASTICITY_ROUTINES::ELASTICITY_PROBLEM_SETUP(), EQUATIONS_
MAPPING_ROUTINES::EQUATIONS_MAPPING_CALCULATE(), EQUATIONS_MAPPING_
ROUTINES::EQUATIONS_MAPPING_CREATE_FINISH(), EQUATIONS_MAPPING_
ROUTINES::EQUATIONS_MAPPING_CREATE_START(), EQUATIONS_MAPPING_
ROUTINES::EQUATIONS_MAPPING_CREATE_VALUES_CACHE_FINALISE(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_CREATE_VALUES_CACHE_INITIALISE(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_DESTROY(), EQUATIONS_
MAPPING_ROUTINES::EQUATIONS_MAPPING_EQUATIONS_JACOBIAN_TO_VARIABLE_
MAP_FINALISE(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_
EQUATIONS_JACOBIAN_TO_VARIABLE_MAP_INITIALISE(), EQUATIONS_MAPPING_
EQUATIONS_JACOBIAN_TO_VARIABLE_MAP_INITIALISE()

```

```

ROUTINES::EQUATIONS_MAPPING_EQUATIONS_MATRIX_TO_VARIABLE_MAP_FINALISE(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_EQUATIONS_MATRIX_-
TO_VARIABLE_MAP_INITIALISE(),           EQUATIONS_MAPPING_ROUTINES::EQUATIONS_-
MAPPING_FINALISE(),           EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_-
INITIALISE(),           EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_LINEAR_-
MAPPING_FINALISE(),           EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_-
LINEAR_MAPPING_INITIALISE(),           EQUATIONS_MAPPING_ROUTINES::EQUATIONS_-
MAPPING_LINEAR_MATRICES_COEFFICIENTS_SET(),           EQUATIONS_MAPPING_-
ROUTINES::EQUATIONS_MAPPING_LINEAR_MATRICES_NUMBER_SET(),           EQUATIONS_-
MAPPING_ROUTINES::EQUATIONS_MAPPING_LINEAR_MATRICES_VARIABLE_-
TYPES_SET(),           EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_-
NONLINEAR_MAPPING_FINALISE(),           EQUATIONS_MAPPING_ROUTINES::EQUATIONS_-
MAPPING_NONLINEAR_MAPPING_INITIALISE(),           EQUATIONS_MAPPING_-
ROUTINES::EQUATIONS_MAPPING_RESIDUAL_COEFFICIENT_SET(),           EQUATIONS_-
MAPPING_ROUTINES::EQUATIONS_MAPPING_RESIDUAL_VARIABLE_TYPE_SET(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_RHS_COEFFICIENT_SET(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_RHS_MAPPING_FINALISE(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_RHS_MAPPING_INITIALISE(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_RHS_VARIABLE_TYPE_SET(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_SOURCE_COEFFICIENT_-
SET(),           EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_SOURCE_MAPPING_-
FINALISE(),           EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_SOURCE_-
MAPPING_INITIALISE(),           EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_-
SOURCE_VARIABLE_TYPE_SET(),           EQUATIONS_MAPPING_ROUTINES::EQUATIONS_-
MAPPING_VARIABLE_TO_EQUATIONS_COLUMN_MAP_FINALISE(),           EQUATIONS_-
MAPPING_ROUTINES::EQUATIONS_MAPPING_VARIABLE_TO_EQUATIONS_JACOBIAN_-
MAP_FINALISE(),           EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_-
VARIABLE_TO_EQUATIONS_JACOBIAN_MAP_INITIALISE(),           EQUATIONS_MAPPING_-
ROUTINES::EQUATIONS_MAPPING_VARIABLE_TO_EQUATIONS_MATRICES_MAP_-
FINALISE(),           EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_VARIABLE_-
TO_EQUATIONS_MATRICES_MAP_INITIALISE(),           F90C::F2CSTRING(),           FINITE_ELEMENT_-
ROUTINES::FEM_ELEMENT_MATRICES_FINALISE(),           FINITE_ELEMENT_ROUTINES::FEM_-
ELEMENT_MATRICES_INITIALISE(),           FIELD_ROUTINES::FI(),           FIELD_IO_ROUTINES::FIE(),
FIELD_ROUTINES::FIELD_COMPONENT_INTER(),           FIELD_ROUTINES::FIELD_COMPONENT_-
INTERPOLATION_GET(),           FIELD_ROUTINES::FIELD_COMPONENT_MESH_COM(),           FIELD_-
ROUTINES::FIELD_COMPONENT_MESH_COMPONENT_GET(),           FIELD_ROUTINES::FIELD_-
CREATE_FINISH(),           FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_FINALISE(),           FIELD_-
ROUTINES::FIELD_CREATE_VALUES_CACHE_INITIALISE(),           FIELD_ROUTINES::FIELD_-
DEPENDENT_TYPE_GET(),           FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET_NUMBER(),
FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET_PTR(),           FIELD_ROUTINES::FIELD_-
DESTROY(),           FIELD_ROUTINES::FIELD_DIMENSION_GET(),           FIELD_ROUTINES::FIELD_-
DIMENSION_SET_NUMBER(),           FIELD_ROUTINES::FIELD_DIMENSION_SET_PTR(),           FIELD_-
ROUTINES::FIELD_INTERPOLATE_GAUSS(),           FIELD_ROUTINES::FIELD_INTERPOLATE_XI(),
FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_FINALISE(),           FIELD_ROUTINES::FIELD_-
INTERPOLATED_POINT_INITIALISE(),           FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_-
METRICS_CALCULATE(),           FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_-
FINALISE(),           FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_INITIALISE(),
FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(),           FIELD_-
ROUTINES::FIELD_INTERPOLATION_PARAMETERS_FINALISE(),           FIELD_ROUTINES::FIELD_-
INTERPOLATION_PARAMETERS_INITIALISE(),           FIELD_ROUTINES::FIELD_INTERPOLATION_-
PARAMETERS_LINE_GET(),           FIELD_IO_ROUTINES::FIELD_IO_BASIS_LHTP_FAMILY_-
LABEL(),           FIELD_IO_ROUTINES::FIELD_IO_CREATE_FIELDS(),           FIELD_IO_ROUTINES::FIELD_-
IO_DERIVATIVE_INFO(),           FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_-
SET_ATTACH_LOCAL_PROCESS(),           FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_-

```

```

INFO_SET_FINALIZE(),           FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_-
INITIALISE(),     FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_SORT(),   FIELD_-
IO_ROUTINES::FIELD_IO_ELEMENTS_EXPORT(),           FIELD_IO_ROUTINES::FIELD_IO_-
EXPORT_ELEMENTAL_GROUP_HEADER_FORTRAN(),          FIELD_IO_ROUTINES::FIELD_-
IO_EXPORT_ELEMENTS_INTO_(),          FIELD_IO_ROUTINES::FIELD_IO_EXPORT_NODAL_-
GROUP_HEADER_FORTRA(),    FIELD_IO_ROUTINES::FIELD_IO_EXPORT_NODES_INTO_-
LOC(),      FIELD_IO_ROUTINES::FIELD_IO_FIELD_INFO(),   FIELD_IO_ROUTINES::FIELD_-
IO_FILEDS_GROUP_INFO_GET(),          FIELD_IO_ROUTINES::FIELD_IO_FILEDS_IMPORT(), 
FIELD_IO_ROUTINES::FIELD_IO_FILL_BASIS_INFO(),       FIELD_IO_ROUTINES::FIELD_-
IO_FORTRAN_FILE_CLOSE(),          FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_-_
OPEN(),      FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_DP(),   FIELD_IO_-
ROUTINES::FIELD_IO_FORTRAN_FILE_READ_INTG(),        FIELD_IO_ROUTINES::FIELD_-
IO_FORTRAN_FILE_READ_STRING(),       FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_-_
REWIND(),      FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_DP(),   FIELD_IO_-
ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_INTG(),        FIELD_IO_ROUTINES::FIELD_-
IO_FORTRAN_FILE_WRITE_STRING(),      FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_-_
MESH(),      FIELD_IO_ROUTINES::FIELD_IO_LABEL_DERIVATIVE_INFO_GET(),   FIELD_-
IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET(),        FIELD_IO_ROUTINES::FIELD_-
IO_MULTI_FILES_INFO_GET(),         FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_-_
ATTACH_LOCAL_PROCESS(),          FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_-_
FINALIZE(),     FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_INITIALISE(),   FIELD_-
IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_SORT(),        FIELD_IO_ROUTINES::FIELD_-
IO_NODES_EXPORT(),          FIELD_IO_ROUTINES::FIELD_IO_TRANSLATE_LABEL_INTO_-
INTERPOLATION_TYPE(),          FIELD_ROUTINES::FIELD_VARIABLE_COMPONENT_-_
FINALISE(),     FIELD_ROUTINES::FIELD_VARIABLE_COMPONENT_INITIALISE(),   FIELD_-
ROUTINES::FIELD_VARIABLES_FINALISE(),          FIELD_ROUTINES::FIELD_VARIABLES_-_
INITIALISE(),     FIELD_ROUTINES::FIELDS_FINALISE(),        FIELD_ROUTINES::FIELDS_-_
INITIALISE(),      FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_-_
SET_SETUP(),      FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_-_
SET_SUBTYPE_SET(),      FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITYFINITE_-_
ELEMENT_JACOBIAN_EVALUATE(),  FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_-_
FINITE_ELEMENT_RESIDUAL_EVALUATE(),  FINITE_ELASTICITY_ROUTINES::FINITE_-_
ELASTICITY_GAUSS_CAUCHY_TENSOR(),  FINITE_ELASTICITY_ROUTINES::FINITE_-_
ELASTICITY_GAUSS_DEFORMATION_GRADEINT_TENSOR(),  FINITE_ELASTICITY_-_
ROUTINES::FINITE_ELASTICITY_GAUSS_DFDZ(),  FINITE_ELASTICITY_ROUTINES::FINITE_-_
ELASTICITY_GAUSS_DXDNU(),        FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_-_
PROBLEM_SETUP(),          FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_-_
PROBLEM_SUBTYPE_SET(),        GENERATED_MESH_ROUTINES::GENERATED_-_
MESH_BASIS_GET(),          GENERATED_MESH_ROUTINES::GENERATED_MESH_-_
BASIS_SET_NUMBER(),          GENERATED_MESH_ROUTINES::GENERATED_MESH_-_
BASIS_SET_PTR(),          GENERATED_MESH_ROUTINES::GENERATED_MESH_CREATE_-_
FINISH(),          GENERATED_MESH_ROUTINES::GENERATED_MESH_CREATE_-_
START(),          GENERATED_MESH_ROUTINES::GENERATED_MESH_DESTROY_-_
NUMBER(),          GENERATED_MESH_ROUTINES::GENERATED_MESH_DESTROY_-_
PTR(),          GENERATED_MESH_ROUTINES::GENERATED_MESH_EXTENT_GET(), 
GENERATED_MESH_ROUTINES::GENERATED_MESH_EXTENT_SET_NUMBER(), 
GENERATED_MESH_ROUTINES::GENERATED_MESH_EXTENT_SET_PTR(), 
GENERATED_MESH_ROUTINES::GENERATED_MESH_FINALISE(),        GENERATED_-_
MESH_ROUTINES::GENERATED_MESH_INITIALISE(),          GENERATED_MESH_-_
ROUTINES::GENERATED_MESH_NUMBER_OF_ELEMENTS_GET(),        GENERATED_-_
MESH_ROUTINES::GENERATED_MESH_NUMBER_OF_ELEMENTS_SET_NUMBER(), 
GENERATED_MESH_ROUTINES::GENERATED_MESH_NUMBER_OF_ELEMENTS_-_
SET_PTR(),          GENERATED_MESH_ROUTINES::GENERATED_MESH_ORIGIN_-_
GET(),          GENERATED_MESH_ROUTINES::GENERATED_MESH_ORIGIN_SET_-

```

```

NUMBER(),           GENERATED_MESH_ROUTINES::GENERATED_MESH_ORIGIN_SET_-
PTR(),             GENERATED_MESH_ROUTINES::GENERATED_MESH_REGULAR_CREATE_-
FINISH(),           GENERATED_MESH_ROUTINES::GENERATED_MESH_REGULAR_-
FINALISE(),         GENERATED_MESH_ROUTINES::GENERATED_MESH_REGULAR_-
INITIALISE(),       GENERATED_MESH_ROUTINES::GENERATED_MESH_TYPE_GET(),
GENERATED_MESH_ROUTINES::GENERATED_MESH_TYPE_SET_NUMBER(),
GENERATED_MESH_ROUTINES::GENERATED_MESH_TYPE_SET_PTR(),      GENERATED_-
MESH_ROUTINES::GENERATED_MESH_USER_NUMBER_FIND(),            GENERATED_-
MESH_ROUTINES::GENERATED_MESHES_FINALISE(),                  GENERATED_MESH_-
ROUTINES::GENERATED_MESHES_INITIALISE(),                     SORTING::HEAP_SORT_DP(),
SORTING::HEAP_SORT_INTG(),        SORTING::HEAP_SORT_SP(),   BINARY_FILE::INQUIRE_-
EOF_BINARY_FILE(), MATHS::INVERT_FULL_DP(), MATHS::INVERT_FULL_SP(), LAPLACE_-
EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_CALCULATE(), LAPLACE_-
EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_PARAMETER_SET_UPDATE(),
LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUIVATIONS_SET_SETUP(),
LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUIVATIONS_SET_STANDARD_-
SETUP(),           LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUIVATIONS_SET_-
SUBTYPE_SET(),      LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATIONFINITE_-
ELEMENT_CALCULATE(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-
FIXED_CONDITIONS_(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-
INTE(),             LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SETUP(),
LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(),
LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SUBTYPE_SET(),
LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUIVATIONS_SET_SETUP(),
LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUIVATIONS_SET_SUBTYPE_-
SET(),             LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITYFINITE_ELEMENT_-
CALCULATE(),        LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_-
SETUP(),           LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SUBTYPE_-
SET(),             LISTS::LIST_CREATE_FINISH(),   LISTS::LIST_CREATE_START(),   LISTS::LIST_-
DATA_TYPE_SET(),    LISTS::LIST_DESTROY(),      LISTS::LIST_DETACH_AND_DESTROY_DP(),
LISTS::LIST_DETACH_AND_DESTROY_INTG(),   LISTS::LIST_DETACH_AND_DESTROY_-
SP(),               LISTS::LIST_FINALISE(),     LISTS::LIST_INITIAL_SIZE_SET(), LISTS::LIST_INITIALISE(),
LISTS::LIST_ITEM_ADD_DP1(),   LISTS::LIST_ITEM_ADD_INTG1(), LISTS::LIST_ITEM_ADD_-
SP1(),              LISTS::LIST_ITEM_DELETE(),  LISTS::LIST_ITEM_IN_LIST_DP1(),  LISTS::LIST_-
ITEM_IN_LIST_INTG1(),  LISTS::LIST_ITEM_IN_LIST_SP1(),   LISTS::LIST_NUMBER_OF_-
ITEMS_GET(),        LISTS::LIST_REMOVE_DUPLICATES(), LISTS::LIST_SEARCH_DP_ARRAY(),
LISTS::LIST_SEARCH_INTG_ARRAY(),        LISTS::LIST_SEARCH_LINEAR_DP_ARRAY(),
LISTS::LIST_SEARCH_LINEAR_INTG_ARRAY(), LISTS::LIST_SEARCH_LINEAR_SP_-
ARRAY(),            LISTS::LIST_SEARCH_SP_ARRAY(),  LISTS::LIST_SORT_BUBBLE_DP_ARRAY(),
LISTS::LIST_SORT_BUBBLE_INTG_ARRAY(),  LISTS::LIST_SORT_BUBBLE_SP_ARRAY(),
LISTS::LIST_SORT_DP_ARRAY(),          LISTS::LIST_SORT_HEAP_DP_ARRAY(), LISTS::LIST_-
SORT_HEAP_INTG_ARRAY(),            LISTS::LIST_SORT_HEAP_SP_ARRAY(), LISTS::LIST_-
SORT_INTG_ARRAY(),              LISTS::LIST_SORT_SHELL_DP_ARRAY(), LISTS::LIST_SORT_-
SHELL_INTG_ARRAY(),            LISTS::LIST_SORT_SHELL_SP_ARRAY(), LISTS::LIST_SORT_-
SP_ARRAY(),            STRINGS::LIST_TO_CHARACTER_C(),  STRINGS::LIST_TO_CHARACTER_-
DP(),                STRINGS::LIST_TO_CHARACTER_INTG(), STRINGS::LIST_TO_CHARACTER_-
L(),                 STRINGS::LIST_TO_CHARACTER_LINTG(), STRINGS::LIST_TO_CHARACTER_-
SP(),                STRINGS::LOGICAL_TO_CHARACTER(),  STRINGS::LOGICAL_TO_VSTRING(),
MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_DP(),   MATRIX_VECTOR::MATRIX_ALL_-
VALUES_SET_INTG(),          MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_L(),  MATRIX_-
VECTOR::MATRIX_ALL_VALUES_SET_SP(),  MATRIX_VECTOR::MATRIX_CREATE_FINISH(),
MATRIX_VECTOR::MATRIX_CREATE_START(),        MATRIX_VECTOR::MATRIX_DATA_GET_-
DP(),                MATRIX_VECTOR::MATRIX_DATA_GET_INTG(), MATRIX_VECTOR::MATRIX_DATA_-
GET_L(),               MATRIX_VECTOR::MATRIX_DATA_GET_SP(),  MATRIX_VECTOR::MATRIX_DATA_-

```

```

TYPE_SET(),      MATRIX_VECTOR::MATRIX_DESTROY(),      MATRIX_VECTOR::MATRIX_-
DUPLICATE(),    MATRIX_VECTOR::MATRIX_FINALISE(),      MATRIX_VECTOR::MATRIX_-
INITIALISE(),   MATRIX_VECTOR::MATRIX_MAX_COLUMNS_PER_ROW_GET(),  MATRIX_-
VECTOR::MATRIX_MAX_SIZE_SET(),          MATRIX_VECTOR::MATRIX_NUMBER_NON_-
ZEROS_SET(),    MATRIX_VECTOR::MATRIX_OUTPUT(),        MATHS::MATRIX_PRODUCT_DP(),
MATHS::MATRIX_PRODUCT_SP(),    MATRIX_VECTOR::MATRIX_SIZE_SET(),      MATRIX_-
VECTOR::MATRIX_STORAGE_LOCATION_FIND(),  MATRIX_VECTOR::MATRIX_STORAGE_-
LOCATIONS_GET(),  MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_SET(),  MATRIX_-
VECTOR::MATRIX_STORAGE_TYPE_GET(),      MATRIX_VECTOR::MATRIX_STORAGE_-
TYPE_SET(),     MATHS::MATRIX_TRANSPOSE_DP(),        MATHS::MATRIX_TRANSPOSE_-
SP(),          MATRIX_VECTOR::MATRIX_VALUES_ADD_DP(),    MATRIX_VECTOR::MATRIX_-
VALUES_ADD_DP1(),   MATRIX_VECTOR::MATRIX_VALUES_ADD_DP2(),    MATRIX_-
VECTOR::MATRIX_VALUES_ADD_INTG(),       MATRIX_VECTOR::MATRIX_VALUES_ADD_-
INTG1(),        MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG2(),   MATRIX_VECTOR::MATRIX_-
VALUES_ADD_L0(),   MATRIX_VECTOR::MATRIX_VALUES_ADD_L10(),    MATRIX_-
VECTOR::MATRIX_VALUES_ADD_L20(),      MATRIX_VECTOR::MATRIX_VALUES_ADD_-
SP(),          MATRIX_VECTOR::MATRIX_VALUES_ADD_SP1(),    MATRIX_VECTOR::MATRIX_-
VALUES_ADD_SP2(),   MATRIX_VECTOR::MATRIX_VALUES_GET_DP(),    MATRIX_-
VECTOR::MATRIX_VALUES_GET_DP10(),     MATRIX_VECTOR::MATRIX_VALUES_GET_-
DP2(),          MATRIX_VECTOR::MATRIX_VALUES_GET_INTG(),   MATRIX_VECTOR::MATRIX_-
VALUES_GET_INTG1(),   MATRIX_VECTOR::MATRIX_VALUES_GET_INTG2(),   MATRIX_-
VECTOR::MATRIX_VALUES_GET_L0(),      MATRIX_VECTOR::MATRIX_VALUES_GET_L10(),
MATRIX_VECTOR::MATRIX_VALUES_GET_L20(),  MATRIX_VECTOR::MATRIX_VALUES_-
GET_SP(),        MATRIX_VECTOR::MATRIX_VALUES_GET_SP1(),    MATRIX_VECTOR::MATRIX_-
VALUES_GET_SP2(),   MATRIX_VECTOR::MATRIX_VALUES_SET_DP(),    MATRIX_-
VECTOR::MATRIX_VALUES_SET_DP1(),     MATRIX_VECTOR::MATRIX_VALUES_SET_-
DP2(),          MATRIX_VECTOR::MATRIX_VALUES_SET_INTG(),   MATRIX_VECTOR::MATRIX_-
VALUES_SET_INTG1(),   MATRIX_VECTOR::MATRIX_VALUES_SET_INTG2(),   MATRIX_-
VECTOR::MATRIX_VALUES_SET_L0(),      MATRIX_VECTOR::MATRIX_VALUES_SET_L10(),
MATRIX_VECTOR::MATRIX_VALUES_SET_L20(),  MATRIX_VECTOR::MATRIX_VALUES_-
SET_SP(),        MATRIX_VECTOR::MATRIX_VALUES_SET_SP1(),    MATRIX_VECTOR::MATRIX_-
VALUES_SET_SP2(),   MESH_ROUTINES::MESH_CREATE_FINISH(),   MESH_ROUTINES::MESH_-
CREATE_START(),   MESH_ROUTINES::MESH_DESTROY(),        MESH_ROUTINES::MESH_-
FINALISE(),      MESH_ROUTINES::MESH_INITIALISE(),   MESH_ROUTINES::MESH_NUMBER_-
OF_COMPONENTS_GET(),  MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_-
SET_NUMBER(),    MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET_PTR(),
MESH_ROUTINES::MESH_NUMBER_OF_ELEMENTS_GET(),   MESH_ROUTINES::MESH_-
NUMBER_OF_ELEMENTS_SET_NUMBER(),   MESH_ROUTINES::MESH_NUMBER_OF_-
ELEMENTS_SET_PTR(),   MESH_ROUTINES::MESH_TOPOLOGY_CALCULATE(),  MESH_-
ROUTINES::MESH_TOPOLOGY_DOFS_CALCULATE(),   MESH_ROUTINES::MESH_-
TOPOLOGY_DOFS_FINALISE(),  MESH_ROUTINES::MESH_TOPOLOGY_DOFS_INITIALISE(),
MESH_ROUTINES::MESH_TOPOLOGY_ELEMENT_FINALISE(),  MESH_ROUTINES::MESH_-
TOPOLOGY_ELEMENT_INITIALISE(),  MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_-
ADJACENT_ELEMENTS_CALCULATE(),   MESH_ROUTINES::MESH_TOPOLOGY_-_
ELEMENTS_BASIS_SET(),   MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_-_
FINISH(),        MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_START(),  MESH_-
ROUTINES::MESH_TOPOLOGY_ELEMENTS_DESTROY(),   MESH_ROUTINES::MESH_-
TOPOLOGY_ELEMENTS_ELEMENT_BASIS_GET(),   MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_-_
ELEMENT_NODES_SET(),   MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_-_
NODES_SET(),      MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_FINALISE(),  MESH_-
ROUTINES::MESH_TOPOLOGY_ELEMENTS_INITIALISE(),   MESH_ROUTINES::MESH_-
TOPOLOGY_ELEMENTS_NUMBER_GET(),   MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_-
NUMBER_SET(),    MESH_ROUTINES::MESH_TOPOLOGY_FINALISE(),   MESH_-

```

```

ROUTINES::MESH_TOPOLOGY_INITIALISE(),      MESH_ROUTINES::MESH_TOPOLOGY_-
NODE_FINALISE(),   MESH_ROUTINES::MESH_TOPOLOGY_NODE_INITIALISE(),  MESH_-
ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE(),    MESH_ROUTINES::MESH_-
TOPOLOGY_NODES_DERIVATIVES_CALCULATE(),      MESH_ROUTINES::MESH_-
TOPOLOGY_NODES_FINALISE(),      MESH_ROUTINES::MESH_TOPOLOGY_NODES_-
INITIALISE(),      MESH_ROUTINES::MESH_TOPOLOGY_NODES_SURROUNDING_-
ELEMENTS_CALCULATE(),  MESH_ROUTINES::MESH_USER_NUMBER_FIND(),  MESH_-
ROUTINES::MESSES_FINALISE(),   MESH_ROUTINES::MESSES_INITIALISE(),  CMISS_-
MPI::MPI_ERROR_CHECK(),      NODE_ROUTINES::NODE_CHECK_EXISTS(),  NODE_-
ROUTINES::NODE_DESTROY(),     NODE_ROUTINES::NODE_INITIAL_POSITION_SET(), NODE_-
ROUTINES::NODE_NUMBER_SET(),  NODE_ROUTINES::NODES_CREATE_FINISH(), NODE_-
ROUTINES::NODES_CREATE_START(), NODE_ROUTINES::NODES_FINALISE(), NODE_-
ROUTINES::NODES_INITIALISE(), MATHS::NORMALISE_DP(),  MATHS::NORMALISE_SP(), 
STRINGS::NUMBER_TO_CHARACTER_DP(),  STRINGS::NUMBER_TO_CHARACTER_INTG(), 
STRINGS::NUMBER_TO_CHARACTER_LINTG(),  STRINGS::NUMBER_TO_CHARACTER_-
SP(),  STRINGS::NUMBER_TO_VSTRING_DP(),  STRINGS::NUMBER_TO_VSTRING_INTG(), 
STRINGS::NUMBER_TO_VSTRING_LINTG(),  STRINGS::NUMBER_TO_VSTRING_SP(), 
BINARY_FILE::OPEN_BINARY_FILE(),  BINARY_FILE::OPEN_CMISS_BINARY_FILE(), CMISS_-
PARMETIS::PARMETIS_PARTKWAY(),  CMISS_PARMETIS::PARMETIS_PARTMESHKWAY(), 
CMISS_PETSC::PETSC_ERRORHANDLING_SET_OFF(),  CMISS_PETSC::PETSC_-
ERRORHANDLING_SET_ON(),  CMISS_PETSC::PETSC_FINALIZE(),  CMISS_PETSC::PETSC_-
INITIALIZE(),  CMISS_PETSC::PETSC_ISCOLORINGDESTROY(),  CMISS_PETSC::PETSC_-
ISCOLORINGFINALISE(),  CMISS_PETSC::PETSC_ISCOLORINGINITIALISE(),  CMISS_-
PETSC::PETSC_ISDESTROY(),  CMISS_PETSC::PETSC_ISFINALISE(),  CMISS_PETSC::PETSC_-
ISINITIALISE(),  CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGAPPLY(), 
CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGAPPLYIS(),  CMISS_-
PETSC::PETSC_ISLOCALTOGLOBALMAPPINGCREATE(),  CMISS_PETSC::PETSC_-
ISLOCALTOGLOBALMAPPINGDESTROY(),  CMISS_PETSC::PETSC_-
ISLOCALTOGLOBALMAPPINGFINALISE(),  CMISS_PETSC::PETSC_-
ISLOCALTOGLOBALMAPPINGINITIALISE(),  CMISS_PETSC::PETSC_KSPCREATE(), 
CMISS_PETSC::PETSC_KSPDESTROY(),  CMISS_PETSC::PETSC_KSPFINALISE(), 
CMISS_PETSC::PETSC_KSPGETCONVERGEDREASON(),  CMISS_PETSC::PETSC_-
KSPGETITERATIONNUMBER(),  CMISS_PETSC::PETSC_KSPGETPC(),  CMISS_-
PETSC::PETSC_KSPGETRESIDUALNORM(),  CMISS_PETSC::PETSC_KSPINITIALISE(), 
CMISS_PETSC::PETSC_KSPSETFROMOPTIONS(),  CMISS_PETSC::PETSC_-
KSPSETOPERATORS(),  CMISS_PETSC::PETSC_KSPSETTOLERANCES(),  CMISS_-
PETSC::PETSC_KSPSETTYPE(),  CMISS_PETSC::PETSC_KSPSETUP(),  CMISS_PETSC::PETSC_-
KSPSOLVE(),  CMISS_PETSC::PETSC_LOGPRINTSUMMARY(),  CMISS_PETSC::PETSC_-
MATASSEMBLYBEGIN(),  CMISS_PETSC::PETSC_MATASSEMBLYEND(),  CMISS_-
PETSC::PETSC_MATCREATE(),  CMISS_PETSC::PETSC_MATCREATEMPIAIJ(),  CMISS_-
PETSC::PETSC_MATCREATEMPIENSE(),  CMISS_PETSC::PETSC_MATCREATESEQAIJ(), 
CMISS_PETSC::PETSC_MATCREATESEQDENSE(),  CMISS_PETSC::PETSC_MATDESTROY(), 
CMISS_PETSC::PETSC_MATFDCOLORINGCREATE(),  CMISS_PETSC::PETSC_-
MATFDCOLORINGDESTROY(),  CMISS_PETSC::PETSC_MATFDCOLORINGFINALISE(), 
CMISS_PETSC::PETSC_MATFDCOLORINGINITIALISE(),  CMISS_PETSC::PETSC_-
MATFDCOLORINGSETFROMOPTIONS(),  CMISS_PETSC::PETSC_MATFINALISE(), 
CMISS_PETSC::PETSC_MATGETARRAY(),  CMISS_PETSC::PETSC_MATGETCOLORING(), 
CMISS_PETSC::PETSC_MATGETOWNERSHIPRANGE(),  CMISS_PETSC::PETSC_-
MATGETVALUES(),  CMISS_PETSC::PETSC_MATINITIALISE(),  CMISS_PETSC::PETSC_-
MATRESTOREARRAY(),  CMISS_PETSC::PETSC_MATSETLOCALTOGLOBALMAPPING(), 
CMISS_PETSC::PETSC_MATSETOPTION(),  CMISS_PETSC::PETSC_MATSETSIZES(),  CMISS_-
PETSC::PETSC_MATSETVALUE(),  CMISS_PETSC::PETSC_MATSETVALUELOCAL(),  CMISS_-
PETSC::PETSC_MATSETVALUES(),  CMISS_PETSC::PETSC_MATSETVALUESLOCAL(), 
CMISS_PETSC::PETSC_MATVIEW(),  CMISS_PETSC::PETSC_MATZEROENTRIES(),

```

CMISS_PETSC::PETSC_PCFINALISE(), CMISS_PETSC::PETSC_PCINITIALISE(),
 CMISS_PETSC::PETSC_PCSETTYPE(), CMISS_PETSC::PETSC_SNESCREATE(),
 CMISS_PETSC::PETSC_SNESDESTROY(), CMISS_PETSC::PETSC_SNESFINALISE(),
 CMISS_PETSC::PETSC_SNESGETCONVERGEDREASON(), CMISS_PETSC::PETSC_SNESGETFUNCTIONNORM(), CMISS_PETSC::PETSC_SNESGETITERATIONNUMBER(),
 CMISS_PETSC::PETSC_SNESINITIALISE(), CMISS_PETSC::PETSC_SNESLINESEARCHSET(),
 CMISS_PETSC::PETSC_SNESLINESEARCHSETPARAMS(), CMISS_PETSC::PETSC_SNESSETFROMOPTIONS(), CMISS_PETSC::PETSC_SNESSETFUNCTION(), CMISS_PETSC::PETSC_SNESSETJACOBIAN_MATFDCOLORING(), CMISS_PETSC::PETSC_SNESSETJACOBIAN_SOLVER(), CMISS_PETSC::PETSC_SNESSETTOLERANCES(),
 CMISS_PETSC::PETSC_SNESSETTRUSTREGIONTOLERANCE(), CMISS_PETSC::PETSC_SNESSETTYPE(), CMISS_PETSC::PETSC_SNESSOLVE(), CMISS_PETSC::PETSC_VECASSEMBLYBEGIN(), CMISS_PETSC::PETSC_VECASSEMBLYEND(), CMISS_PETSC::PETSC_VECCREATE(), CMISS_PETSC::PETSC_VECCREATEGHOST(), CMISS_PETSC::PETSC_VECCREATEGHOSTWITHARRAY(), CMISS_PETSC::PETSC_VECCREATEMPI(), CMISS_PETSC::PETSC_VECCREATEMPIWITHARRAY(), CMISS_PETSC::PETSC_VECCREATESEQ(), CMISS_PETSC::PETSC_VECCREATESEQWITHARRAY(), CMISS_PETSC::PETSC_VECDESTROY(), CMISS_PETSC::PETSC_VECDUPLICATE(), CMISS_PETSC::PETSC_VECFINALISE(), CMISS_PETSC::PETSC_VECGETARRAY(), CMISS_PETSC::PETSC_VECGETARRAYF90(), CMISS_PETSC::PETSC_VECGETLOCALSIZE(), CMISS_PETSC::PETSC_VECGETOWNERSHIPRANGE(), CMISS_PETSC::PETSC_VECGETSIZE(), CMISS_PETSC::PETSC_VECGETVALUES(), CMISS_PETSC::PETSC_VECHOSTGETLOCALFORM(), CMISS_PETSC::PETSC_VECHOSTRESTORELOCALFORM(), CMISS_PETSC::PETSC_VECHOSTUPDATEBEGIN(), CMISS_PETSC::PETSC_VECHOSTUPDATEEND(), CMISS_PETSC::PETSC_VECINITIALISE(), CMISS_PETSC::PETSC_VECRESTOREARRAY(), CMISS_PETSC::PETSC_VECRESTOREARRAYF90(), CMISS_PETSC::PETSC_VECSET(), CMISS_PETSC::PETSC_VECSETFROMOPTIONS(), CMISS_PETSC::PETSC_VECSETLOCALTOGLOBALMAPPING(), CMISS_PETSC::PETSC_VECSETSIZES(), CMISS_PETSC::PETSC_VECSETVALUES(), CMISS_PETSC::PETSC_VECSETVALUESLOCAL(), CMISS_PETSC::PETSC_VECVIEW(), PROBLEM_ROUTINES::PROBLEM_CONTROL_CREATE_FINISH(), PROBLEM_ROUTINES::PROBLEM_CONTROL_CREATE_START(), PROBLEM_ROUTINES::PROBLEM_CONTROL_DESTROY(), PROBLEM_ROUTINES::PROBLEM_CONTROL_FINALISE(), PROBLEM_ROUTINES::PROBLEM_CONTROL_INITIALISE(), PROBLEM_ROUTINES::PROBLEM_CREATE_FINISH(), PROBLEM_ROUTINES::PROBLEM_CREATE_START(), PROBLEM_ROUTINES::PROBLEM_DESTROY_NUMBER(), PROBLEM_ROUTINES::PROBLEM_DESTROY_PTR(), PROBLEM_ROUTINES::PROBLEM_FINALISE(), PROBLEM_ROUTINES::PROBLEM_INITIALISE(), PROBLEM_ROUTINES::PROBLEM_SETUP(), PROBLEM_ROUTINES::PROBLEM SOLUTION EQUATIONS SET ADD(), PROBLEM_ROUTINES::PROBLEM SOLUTION FINALISE(), PROBLEM_ROUTINES::PROBLEM SOLUTION INITIALISE(), PROBLEM_ROUTINES::PROBLEM SOLUTION JACOBIAN EVALUATE(), PROBLEM_ROUTINES::PROBLEM SOLUTION RESIDUAL EVALUATE(), PROBLEM_ROUTINES::PROBLEM SOLUTION SOLVE(), PROBLEM_ROUTINES::PROBLEM SOLUTIONS CREATE FINISH(), PROBLEM_ROUTINES::PROBLEM SOLUTIONS CREATE START(), PROBLEM_ROUTINES::PROBLEM SOLUTIONS FINALISE(), PROBLEM_ROUTINES::PROBLEM SOLUTIONS INITIALISE(), PROBLEM_ROUTINES::PROBLEM SOLVE(), PROBLEM_ROUTINES::PROBLEM SOLVER CREATE FINISH(), PROBLEM_ROUTINES::PROBLEM SOLVER CREATE START(), PROBLEM_ROUTINES::PROBLEM SOLVER DESTROY(), PROBLEM_ROUTINES::PROBLEM SOLVER GET(), PROBLEM_ROUTINES::PROBLEM SPECIFICATION GET NUMBER(), PROBLEM_ROUTINES::PROBLEM SPECIFICATION GET PTR(), PROBLEM_ROUTINES::PROBLEM SPECIFICATION SET NUMBER(), PROBLEM_ROUTINES::PROBLEM SPECIFICATION SET PTR(), PROBLEM_ROUTINES::PROBLEM USER NUMBER FIND(), PROBLEM_ROUTINES::PROBLEMS FINALISE(), PROBLEM_ROUTINES::PROBLEMS INITIALISE(), BINARY FILE::READ -

```

BINARY_FILE_CHARACTER(),      BINARY_FILE::READ_BINARY_FILE_DP(),      BINARY_-
FILE::READ_BINARY_FILE_DP1(),  BINARY_FILE::READ_BINARY_FILE_DPC(),  BINARY_-
FILE::READ_BINARY_FILE_DPC1(), BINARY_FILE::READ_BINARY_FILE_INTG(), BINARY_-
FILE::READ_BINARY_FILE_INTG1(), BINARY_FILE::READ_BINARY_FILE_LINTG(), BINARY_-
FILE::READ_BINARY_FILE_LINTG1(),   BINARY_FILE::READ_BINARY_FILE_LOGICAL(), 
BINARY_FILE::READ_BINARY_FILE_LOGICAL1(),      BINARY_FILE::READ_BINARY_-
FILE_SINTG(),      BINARY_FILE::READ_BINARY_FILE_SINTG1(),      BINARY_FILE::READ_-
BINARY_FILE_SP(),      BINARY_FILE::READ_BINARY_FILE_SP1(),      BINARY_FILE::READ_-
BINARY_FILE_SPC(),      BINARY_FILE::READ_BINARY_FILE_SPC1(),      BINARY_FILE::READ_-
BINARY_TAG_HEADER(),      REGION_ROUTINES::REGION_COORDINATE_SYSTEM_GET(), 
REGION_ROUTINES::REGION_COORDINATE_SYSTEM_SET_NUMBER(),      REGION_-
ROUTINES::REGION_COORDINATE_SYSTEM_SET_PTR(),      REGION_ROUTINES::REGION_-
CREATE_FINISH(),      REGION_ROUTINES::REGION_CREATE_START(),      REGION_-
ROUTINES::REGION_DESTROY(),  REGION_ROUTINES::REGION_LABEL_GET(),  REGION_-
ROUTINES::REGION_LABEL_SET_NUMBER(),      REGION_ROUTINES::REGION_LABEL_-
SET_PTR(),      REGION_ROUTINES::REGION_SUB_REGION_CREATE_FINISH(),  REGION_-
ROUTINES::REGION_SUB_REGION_CREATE_START(),      REGION_ROUTINES::REGION_-
USER_NUMBER_FIND(),      REGION_ROUTINES::REGION_USER_NUMBER_FIND_PTR(), 
REGION_ROUTINES::REGIONS_FINALISE(),      REGION_ROUTINES::REGIONS_INITIALISE(), 
BINARY_FILE::RESET_BINARY_NUMBER_TAGS(),      BINARY_FILE::SET_BINARY_FILE(), 
SORTING::SHELL_SORT_DP(),  SORTING::SHELL_SORT_INTG(),  SORTING::SHELL_SORT_SP(), 
BINARY_FILE::SKIP_BINARY_FILE(),      BINARY_FILE::SKIP_BINARY_TAGS(),      BINARY_-
FILE::SKIP_CM_BINARY_HEADER(),      MATHS::SOLVE_SMALL_LINEAR_SYSTEM_DP(), 
MATHS::SOLVE_SMALL_LINEAR_SYSTEM_SP(),  SOLVER_ROUTINES::SOLVER_CREATE_-_
FINISH(),  SOLVER_ROUTINES::SOLVER_CREATE_START(),  SOLVER_ROUTINES::SOLVER_-_
DESTROY(),  SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_CREATE_FINISH(),  SOLVER_-_
ROUTINES::SOLVER_EIGENPROBLEM_FINALISE(),      SOLVER_ROUTINES::SOLVER_-_
EIGENPROBLEM_INITIALISE(),      SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_-_
SOLVE(),  SOLVER_ROUTINES::SOLVER_FINALISE(),  SOLVER_ROUTINES::SOLVER_-_
INITIALISE(),  SOLVER_ROUTINES::SOLVER_LIBRARY_SET(),  SOLVER_ROUTINES::SOLVER_-_
LINEAR_CREATE_FINISH(),      SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_CREATE_-_
FINISH(),  SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_FINALISE(),  SOLVER_-_
ROUTINES::SOLVER_LINEAR_DIRECT_INITIALISE(),  SOLVER_ROUTINES::SOLVER_-_
LINEAR_DIRECT_SOLVE(),  SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_TYPE_SET(), 
SOLVER_ROUTINES::SOLVER_LINEAR_FINALISE(),  SOLVER_ROUTINES::SOLVER_-_
LINEAR_INITIALISE(),  SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_ABSOLUTE_-_
TOLERANCE_SET(),  SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_-_
FINISH(),  SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_DIVergence_TOLERANCE_-_
SET(),  SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_FINALISE(),  SOLVER_-_
ROUTINES::SOLVER_LINEAR_ITERATIVE_INITIALISE(),  SOLVER_ROUTINES::SOLVER_-_
LINEAR_ITERATIVE_MAXIMUM_ITERATIONS_SET(),  SOLVER_ROUTINES::SOLVER_-_
LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET(),  SOLVER_ROUTINES::SOLVER_-_
LINEAR_ITERATIVE_RELATIVE_TOLERANCE_SET(),  SOLVER_ROUTINES::SOLVER_-_
LINEAR_ITERATIVE_SOLVE(),  SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_-_
SET(),  SOLVER_ROUTINES::SOLVER_LINEAR_SOLVE(),  SOLVER_ROUTINES::SOLVER_-_
LINEAR_TYPE_SET(),  SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE(), 
SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_FINISH(),  SOLVER_-_
MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_START(),  SOLVER_MATRICES_-_
ROUTINES::SOLVER_MATRICES_DESTROY(),  SOLVER_MATRICES_ROUTINES::SOLVER_-_
MATRICES_FINALISE(),  SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_INITIALISE(), 
SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_LIBRARY_TYPE_GET(),  SOLVER_-_
MATRICES_ROUTINES::SOLVER_MATRICES_LIBRARY_TYPE_SET(),  SOLVER_MATRICES_-_
ROUTINES::SOLVER_MATRICES_OUTPUT(),  SOLVER_MATRICES_ROUTINES::SOLVER_-_
MATRICES_STORAGE_TYPE_GET(),  SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_-

```

STORAGE_TYPE_SET(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_FINALISE(),
 SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_FORM(), SOLVER_MATRICES_-
 ROUTINES::SOLVER_MATRIX_INITIALISE(), SOLVER_MATRICES_ROUTINES::SOLVER_-
 MATRIX_STRUCTURE_CALCULATE(), SOLVER_ROUTINES::SOLVER_NONLINEAR_-
 ABSOLUTE_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_-
 CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_NONLINEAR_FINALISE(), SOLVER_-
 ROUTINES::SOLVER_NONLINEAR_INITIALISE(), SOLVER_ROUTINES::SOLVER_-
 NONLINEAR_JACOBIAN_EVALUATE(), SOLVER_ROUTINES::SOLVER_NONLINEAR_-
 LINESEARCH_ALPHA_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_-
 CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_-
 FINALISE(), SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_INITIALISE(),
 SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_MAXSTEP_SET(), SOLVER_-
 ROUTINES::SOLVER_NONLINEAR_LINESEARCH_SOLVE(), SOLVER_ROUTINES::SOLVER_-
 NONLINEAR_LINESEARCH_STEPTOL_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_-
 LINESEARCH_TYPE_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_MAXIMUM_-
 FUNCTION_EVALUATIONS_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_-
 MAXIMUM_ITERATIONS_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_-
 RELATIVE_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_RESIDUAL_-
 EVALUATE(), SOLVER_ROUTINES::SOLVER_NONLINEAR SOLUTION_TOLERANCE_SET(),
 SOLVER_ROUTINES::SOLVER_NONLINEAR_SOLVE(), SOLVER_ROUTINES::SOLVER_-
 NONLINEAR_TRUSTREGION_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_-
 NONLINEAR_TRUSTREGION_DELTA0_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_-
 TRUSTREGION_FINALISE(), SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_-
 INITIALISE(), SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_SOLVE(),
 SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_TOLERANCE_SET(), SOLVER_-
 ROUTINES::SOLVER_NONLINEAR_TYPE_SET(), SOLVER_ROUTINES::SOLVER_OUTPUT_-
 TYPE_SET(), SOLVER_ROUTINES::SOLVER_SOLVE(), SOLVER_ROUTINES::SOLVER_-
 SPARSITY_TYPE_SET(), SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_-
 CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_FINALISE(),
 SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_INITIALISE(), SOLVER_-
 ROUTINES::SOLVER_TIME_INTEGRATION_SOLVE(), SOLVER_ROUTINES::SOLVER_-
 VARIABLES_UPDATE(), STRINGS::STRING_TO_DOUBLE_C(), STRINGS::STRING_-
 TO_DOUBLE_VS(), STRINGS::STRING_TO_INTEGER_C(), STRINGS::STRING_TO_-
 INTEGER_VS(), STRINGS::STRING_TO_LOGICAL_C(), STRINGS::STRING_TO_LOGICAL_-
 VS(), STRINGS::STRING_TO_LONG_INTEGER_C(), STRINGS::STRING_TO_LONG_-
 INTEGER_VS(), FIELD_IO_ROUTINES::STRING_TO_MUTI_INTEGERS_VS(), FIELD_-
 IO_ROUTINES::STRING_TO_MUTI_REALS_VS(), STRINGS::STRING_TO_SINGLE_C(),
 STRINGS::STRING_TO_SINGLE_VS(), TREES::TREE_CREATE_FINISH(), TREES::TREE_-
 CREATE_START(), TREES::TREE_DESTROY(), TREES::TREE_DETACH_AND_DESTROY(),
 TREES::TREE_DETACH_IN_ORDER(), TREES::TREE_FINALISE(), TREES::TREE_INITIALISE(),
 TREES::TREE_INSERT_TYPE_SET(), TREES::TREE_ITEM_DELETE(), TREES::TREE_-
 ITEM_INSERT(), TREES::TREE_NODE_FINALISE(), TREES::TREE_NODE_INITIALISE(),
 TREES::TREE_NODE_KEY_GET(), TREES::TREE_NODE_VALUE_GET(), TREES::TREE_-
 NODE_VALUE_SET(), TREES::TREE_OUTPUT(), TREES::TREE_OUTPUT_IN_ORDER(),
 TREES::TREE_PREDECESSOR(), TREES::TREE_SEARCH(), TREES::TREE_SUCCESSOR(),
 MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_DP(), MATRIX_VECTOR::VECTOR_ALL_-
 VALUES_SET_INTG(), MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_L(), MATRIX_-
 VECTOR::VECTOR_ALL_VALUES_SET_SP(), MATRIX_VECTOR::VECTOR_CREATE_FINISH(),
 MATRIX_VECTOR::VECTOR_CREATE_START(), MATRIX_VECTOR::VECTOR_DATA_GET_-
 DP(), MATRIX_VECTOR::VECTOR_DATA_GET_INTG(), MATRIX_VECTOR::VECTOR_DATA_-
 GET_L(), MATRIX_VECTOR::VECTOR_DATA_GET_SP(), MATRIX_VECTOR::VECTOR_DATA_-
 TYPE_SET(), MATRIX_VECTOR::VECTOR_DESTROY(), MATRIX_VECTOR::VECTOR_-
 DUPLICATE(), MATRIX_VECTOR::VECTOR_FINALISE(), MATRIX_VECTOR::VECTOR_-
 INITIALISE(), MATRIX_VECTOR::VECTOR_SIZE_SET(), MATRIX_VECTOR::VECTOR_-

```
VALUES_GET_DP(),      MATRIX_VECTOR::VECTOR_VALUES_GET_DP1(),      MATRIX_-
VECTOR::VECTOR_VALUES_GET_INTG(),      MATRIX_VECTOR::VECTOR_VALUES_GET_-
INTG1(),      MATRIX_VECTOR::VECTOR_VALUES_GET_L(),      MATRIX_VECTOR::VECTOR_-
VALUES_GET_L1(),      MATRIX_VECTOR::VECTOR_VALUES_GET_SP(),      MATRIX_-
VECTOR::VECTOR_VALUES_GET_SP1(),      MATRIX_VECTOR::VECTOR_VALUES_SET_-
DP(),      MATRIX_VECTOR::VECTOR_VALUES_SET_DP1(),      MATRIX_VECTOR::VECTOR_-
VALUES_SET_INTG(),      MATRIX_VECTOR::VECTOR_VALUES_SET_INTG1(),      MATRIX_-
VECTOR::VECTOR_VALUES_SET_L(),      MATRIX_VECTOR::VECTOR_VALUES_SET_L1(), 
MATRIX_VECTOR::VECTOR_VALUES_SET_SP(),      MATRIX_VECTOR::VECTOR_VALUES_-
SET_SP1(),      INPUT_OUTPUT::WR(),      BINARY_FILE::WRITE_BINARY_FILE_CHARACTER(),
BINARY_FILE::WRITE_BINARY_FILE_DP(),      BINARY_FILE::WRITE_BINARY_FILE_DP1(),
BINARY_FILE::WRITE_BINARY_FILE_DPC(),      BINARY_FILE::WRITE_BINARY_FILE_-_
DPC1(),      BINARY_FILE::WRITE_BINARY_FILE_INTG(),      BINARY_FILE::WRITE_BINARY_-
FILE_INTG1(),      BINARY_FILE::WRITE_BINARY_FILE_LINTG(),      BINARY_FILE::WRITE_-
BINARY_FILE_LINTG1(),      BINARY_FILE::WRITE_BINARY_FILE_LOGICAL(),      BINARY_-
FILE::WRITE_BINARY_FILE_LOGICAL1(),      BINARY_FILE::WRITE_BINARY_FILE_SINTG(),
BINARY_FILE::WRITE_BINARY_FILE_SINTG1(),      BINARY_FILE::WRITE_BINARY_FILE_-_
SP(),      BINARY_FILE::WRITE_BINARY_FILE_SP1(),      BINARY_FILE::WRITE_BINARY_FILE_-_
SPC(),      BINARY_FILE::WRITE_BINARY_FILE_SPC1(),      BINARY_FILE::WRITE_BINARY_TAG_-_
HEADER(),      INPUT_OUTPUT::WRITE_STRING_C(),      INPUT_OUTPUT::WRITE_STRING_-_
FMT(),      INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_C(),      INPUT_OUTPUT::WRITE_STRING_-_
FMT_VALUE_DP(),      INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_L(),      INPUT_OUTPUT::WRITE_STRING_-_
FMT_VALUE_LINTG(),      INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_SP(),      INPUT_-_
OUTPUT::WRITE_STRING_FMT_VALUE_VS(),      INPUT_OUTPUT::WRITE_STRING_IDX(),
INPUT_OUTPUT::WRITE_STRING_MATRIX_DP(),      INPUT_OUTPUT::WRITE_STRING_-_
MATRIX_INTG(),      INPUT_OUTPUT::WRITE_STRING_MATRIX_L(),      INPUT_OUTPUT::WRITE_-
STRING_MATRIX_LINTG(),      INPUT_OUTPUT::WRITE_STRING_MATRIX_SP(),      INPUT_-_
OUTPUT::WRITE_STRING_TWO_VALUE_C_C(),      INPUT_OUTPUT::WRITE_STRING_TWO_-_
VALUE_C_DP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_INTG(),      INPUT_-_
OUTPUT::WRITE_STRING_TWO_VALUE_C_L(),      INPUT_OUTPUT::WRITE_STRING_-_
TWO_VALUE_C_SP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_VS(),      INPUT_-_
OUTPUT::WRITE_STRING_TWO_VALUE_DP_C(),      INPUT_OUTPUT::WRITE_STRING_TWO_-_
VALUE_DP_DP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_INTG(),      INPUT_-_
OUTPUT::WRITE_STRING_TWO_VALUE_DP_L(),      INPUT_OUTPUT::WRITE_STRING_-_
TWO_VALUE_DP_SP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_VS(),      INPUT_-_
OUTPUT::WRITE_STRING_TWO_VALUE_INTG_C(),      INPUT_OUTPUT::WRITE_STRING_-_
TWO_VALUE_INTG_DP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_INTG(),
INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_L(),      INPUT_OUTPUT::WRITE_-
STRING_TWO_VALUE_INTG_SP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_-_
VS(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_C(),      INPUT_OUTPUT::WRITE_-
STRING_TWO_VALUE_L_DP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_INTG(),
INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_L(),      INPUT_OUTPUT::WRITE_STRING_-_
TWO_VALUE_L_SP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_VS(),      INPUT_-_
OUTPUT::WRITE_STRING_TWO_VALUE_SP_C(),      INPUT_OUTPUT::WRITE_STRING_TWO_-_
VALUE_SP_DP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_INTG(),      INPUT_-_
OUTPUT::WRITE_STRING_TWO_VALUE_SP_L(),      INPUT_OUTPUT::WRITE_STRING_-_
TWO_VALUE_SP_SP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_VS(),      INPUT_-_
OUTPUT::WRITE_STRING_TWO_VALUE_VS_C(),      INPUT_OUTPUT::WRITE_STRING_TWO_-_
VALUE_VS_DP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_INTG(),      INPUT_-_
OUTPUT::WRITE_STRING_TWO_VALUE_VS_L(),      INPUT_OUTPUT::WRITE_STRING_-_
TWO_VALUE_VS_SP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_VS(),      INPUT_-_
OUTPUT::WRITE_STRING_VALUE_C(),      INPUT_OUTPUT::WRITE_STRING_VALUE_DP(),
INPUT_OUTPUT::WRITE_STRING_VALUE_INTG(),      INPUT_OUTPUT::WRITE_STRING_-
```

VALUE_L(), INPUT_OUTPUT::WRITE_STRING_VALUE_LINTG(), INPUT_OUTPUT::WRITE_STRING_VALUE_SP(), INPUT_OUTPUT::WRITE_STRING_VALUE_VS(), and INPUT_OUTPUT::WRITE_STRING_VS().

6.2.2.7 subroutine BASE_ROUTINES::EXITS (CHARACTER(LEN=*)**,intent(in)** NAME)

Records the exit out of the named procedure.

See also:

[BASE_ROUTINES::ENTERS](#)

Parameters:

NAME The name of the routine exiting

Definition at line 356 of file base_routines.f90.

Referenced by SORTING::BUBBLE_SORT_DP(), SORTING::BUBBLE_SORT_INTG(), SORTING::BUBBLE_SORT_SP(), F90C::C2FSTRING(), CLASSICAL_FIELD_ROUTINES::CL(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SET_SETUP(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELDFINITE_ELEMENT_CALCULATE(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELDFINITE_ELEMENT_JACOBIAN_EVALUATE(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELDFINITE_ELEMENT_RESIDUAL_EVALUATE(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_GET(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_SET(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_SETUP(), BINARY_FILE::CLOSE_BINARY_FILE(), BINARY_FILE::CLOSE_CMISS_BINARYFILE(), COORDINATE_ROUTINES::CO(), COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_TYPE_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_TYPE_INITIALISE(), NUMBER_GET(), NUMBER_GET(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_SP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_SP(), COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE_DP(), COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_NORM(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_PARAMETERS_ADJUST(), COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_FINISH(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_START(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET_NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_GET(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET_NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_NORMAL_SET_NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_NORMAL_SET_PTR()

```

ROUTINES::COORDINATE_SYSTEM_ORIENTATION_SET_NUMBER(),           COORDINATE_-
ROUTINES::COORDINATE_SYSTEM_ORIENTATION_SET_PTR(),             COORDINATE_-
ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET_NUMBER(),               COORDINATE_-
ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET_PTR(),                 COORDINATE_-
ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_NUMBER(), COORDINATE_-
COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_-_
TYPE_SET_PTR(),          COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_-_
NUMBER(),                COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_PTR(),_
COORDINATE_ROUTINES::COORDINATE_SYSTEM_USER_NUMBER_FIND(),   COORDINATE_-
ROUTINES::COORDINATE_SYSTEMS_FINALISE(), COORDINATE_ROUTINES::COORDINATE_-_
SYSTEMS_INITIALISE(),      TIMER::CPU_TIMER(),        MATHS::CROSS_PRODUCT_DP(),_
MATHS::CROSS_PRODUCT_INTG(),    MATHS::CROSS_PRODUCT_SP(),   COORDINATE_-_
ROUTINES::D2ZX_DP(),     MATHS::D_CROSS_PRODUCT_DP(),  MATHS::D_CROSS_PRODUCT_-_
INTG(),      MATHS::D_CROSS_PRODUCT_SP(),    MESH_ROUTINES::DECOMPOSITION_-_
CREATE_FINISH(),       MESH_ROUTINES::DECOMPOSITION_CREATE_START(), MESH_-_
ROUTINES::DECOMPOSITION_DESTROY(),      MESH_ROUTINES::DECOMPOSITION_-_
ELEMENT_DOMAIN_CALCULATE(), MESH_ROUTINES::DECOMPOSITION_ELEMENT_-_
DOMAIN_GET(),            MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_-_
SET(),                  MESH_ROUTINES::DECOMPOSITION_MESH_COMPONENT_NUMBER_-_
GET(),                  MESH_ROUTINES::DECOMPOSITION_MESH_COMPONENT_NUMBER_SET(),_
MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_GET(),        MESH_-_
ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(),   MATHS::DETERMINANT_-_
FULL_DP(),      MATHS::DETERMINANT_FULL_INTG(),  MATHS::DETERMINANT_FULL_-_
SP(),      DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_ADJACENT_DOMAIN_FINALISE(),_
DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_ADJACENT_DOMAIN_INITIALISE(), DOMAIN_-_
MAPPINGS::DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE(),   DOMAIN_-_
MAPPINGS::DOMAIN_MAPPINGS_MAPPING_FINALISE(),  DOMAIN_MAPPINGS::DOMAIN_-_
MAPPINGS_MAPPING_GLOBAL_FINALISE(),  DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_-_
MAPPING_GLOBAL_INITIALISE(), DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_-_
INITIALISE(), MESH_ROUTINES::DOMAIN_MAPPINGS_NODES_FINALISE(), MESH_-_
ROUTINES::DOMAIN_MAPPINGS_NODES_INITIALISE(), MESH_ROUTINES::DOMAIN_-_
TOPOLOGY_CALCULATE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_DOFS_FINALISE(),_
MESH_ROUTINES::DOMAIN_TOPOLOGY_DOFS_INITIALISE(), MESH_ROUTINES::DOMAIN_-_
TOPOLOGY_ELEMENT_FINALISE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_-_
ELEMENT_INITIALISE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_-_
FINALISE(),          MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_INITIALISE(),_
MESH_ROUTINES::DOMAIN_TOPOLOGY_FINALISE(),      MESH_ROUTINES::DOMAIN_-_
TOPOLOGY_INITIALISE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_INITIALISE_-_
FROM_MESH(),      MESH_ROUTINES::DOMAIN_TOPOLOGY_LINE_FINALISE(), MESH_-_
ROUTINES::DOMAIN_TOPOLOGY_LINE_INITIALISE(), MESH_ROUTINES::DOMAIN_-_
TOPOLOGY_LINES_FINALISE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_LINES_-_
INITIALISE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_NODE_FINALISE(), MESH_-_
ROUTINES::DOMAIN_TOPOLOGY_NODE_INITIALISE(), MESH_ROUTINES::DOMAIN_-_
TOPOLOGY_NODES_FINALISE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_-_
INITIALISE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_SURROUNDING_-_
ELEMENTS_CALCULATE(), COORDINATE_ROUTINES::DXZ_DP(), COORDINATE_-_
ROUTINES::DZX_DP(),     MATHS::EIGENVALUE_FULL_DP(),  MATHS::EIGENVALUE_-_
FULL_SP(),      MATHS::EIGENVECTOR_FULL_DP(),    MATHS::EIGENVECTOR_FULL_-_
SP(),      ELASTICITY_ROUTINES::EL(),      ELASTICITY_ROUTINES::ELASTICITY_-_
EQUATIONS_SET_SETUP(), ELASTICITY_ROUTINES::ELASTICITYFINITE_ELEMENT_-_
CALCULATE(),      ELASTICITY_ROUTINES::ELASTICITYFINITE_ELEMENT_JACOBIAN_-_
EVALUATE(),      ELASTICITY_ROUTINES::ELASTICITYFINITE_ELEMENT_RESIDUAL_-_
EVALUATE(),      ELASTICITY_ROUTINES::ELASTICITYPROBLEM_CLASS_TYPE_-_
SET(),      ELASTICITY_ROUTINES::ELASTICITYPROBLEM_SETUP(), EQUATIONS_-

```

```

MAPPING_ROUTINES::EQUATIONS_MAPPING_CALCULATE(),
ROUTINES::EQUATIONS_MAPPING_CREATE_FINISH(),
ROUTINES::EQUATIONS_MAPPING_CREATE_START(),
ROUTINES::EQUATIONS_MAPPING_CREATE_VALUES_CACHE_FINALISE(), EQUATIONS-
MAPPING_ROUTINES::EQUATIONS_MAPPING_CREATE_VALUES_CACHE_INITIALISE(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_DESTROY(), EQUATIONS-
MAPPING_ROUTINES::EQUATIONS_MAPPING_EQUIV_JACOBIAN_TO_VARIABLE-
MAP_FINALISE(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING-
EQUATIONS_JACOBIAN_TO_VARIABLE_MAP_INITIALISE(), EQUATIONS_MAPPING-
ROUTINES::EQUATIONS_MAPPING_EQUIV_MATRIX_TO_VARIABLE_MAP_FINALISE(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_EQUIV_MATRIX_-
TO_VARIABLE_MAP_INITIALISE(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS-
MAPPING_FINALISE(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING-
INITIALISE(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_LINEAR-
MAPPING_FINALISE(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING-
LINEAR_MAPPING_INITIALISE(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS-
MAPPING_LINEAR_MATRICES_COEFFICIENTS_SET(), EQUATIONS_MAPPING-
ROUTINES::EQUATIONS_MAPPING_LINEAR_MATRICES_NUMBER_SET(), EQUATIONS-
MAPPING_ROUTINES::EQUATIONS_MAPPING_LINEAR_MATRICES_VARIABLE-
TYPES_SET(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING-
NONLINEAR_MAPPING_FINALISE(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS-
MAPPING_NONLINEAR_MAPPING_INITIALISE(), EQUATIONS_MAPPING-
ROUTINES::EQUATIONS_MAPPING_RESIDUAL_COEFFICIENT_SET(), EQUATIONS-
MAPPING_ROUTINES::EQUATIONS_MAPPING_RESIDUAL_VARIABLE_TYPE_SET(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_RHS_COEFFICIENT_SET(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_RHS_MAPPING_FINALISE(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_RHS_MAPPING_INITIALISE(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_RHS_VARIABLE_TYPE_SET(),
EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_SOURCE_COEFFICIENT-
SET(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_SOURCE_MAPPING-
FINALISE(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_SOURCE-
MAPPING_INITIALISE(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING-
SOURCE_VARIABLE_TYPE_SET(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS-
MAPPING_VARIABLE_TO_EQUATIONS_COLUMN_MAP_FINALISE(), EQUATIONS-
MAPPING_ROUTINES::EQUATIONS_MAPPING_VARIABLE_TO_EQUATIONS_JACOBIAN-
MAP_FINALISE(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING-
VARIABLE_TO_EQUATIONS_JACOBIAN_MAP_INITIALISE(), EQUATIONS_MAPPING-
ROUTINES::EQUATIONS_MAPPING_VARIABLE_TO_EQUATIONS_MATRICES_MAP_-
FINALISE(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_VARIABLE_-
TO_EQUATIONS_MATRICES_MAP_INITIALISE(), F90C::F2CSTRING(), FINITE_ELEMENT-
ROUTINES::FEM_ELEMENT_MATRICES_FINALISE(), FINITE_ELEMENT_ROUTINES::FEM_-
ELEMENT_MATRICES_INITIALISE(), FIELD_ROUTINES::FI(), FIELD_IO_ROUTINES::FIE(),
FIELD_ROUTINES::FIELD_COMPONENT_INTER(), FIELD_ROUTINES::FIELD_COMPONENT_-
INTERPOLATION_GET(), FIELD_ROUTINES::FIELD_COMPONENT_MESH_COM(), FIELD-
ROUTINES::FIELD_COMPONENT_MESH_COMPONENT_GET(), FIELD_ROUTINES::FIELD_-
CREATE_FINISH(), FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_FINALISE(), FIELD-
ROUTINES::FIELD_CREATE_VALUES_CACHE_INITIALISE(), FIELD_ROUTINES::FIELD_-
DEPENDENT_TYPE_GET(), FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET_NUMBER(),
FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET_PTR(), FIELD_ROUTINES::FIELD_-
DESTROY(), FIELD_ROUTINES::FIELD_DIMENSION_GET(), FIELD_ROUTINES::FIELD_-
DIMENSION_SET_NUMBER(), FIELD_ROUTINES::FIELD_DIMENSION_SET_PTR(), FIELD-
ROUTINES::FIELD_INTERPOLATE_GAUSS(), FIELD_ROUTINES::FIELD_INTERPOLATE_XI(),
FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_FINALISE(), FIELD_ROUTINES::FIELD_-
INTERPOLATED_POINT_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_-

```

METRICS_CALCULATE(), FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_FINALISE(), FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_FINALISE(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), FIELD_IO_ROUTINES::FIELD_IO_BASIS_LHTTP_FAMILY_LABEL(), FIELD_IO_ROUTINES::FIELD_IO_CREATE_FIELDS(), FIELD_IO_ROUTINES::FIELD_IO_DERIVATIVE_INFO(), FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_ATTACH_LOCAL_PROCESS(), FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_FINALIZE(), FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_INITIALISE(), FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_SORT(), FIELD_IO_ROUTINES::FIELD_IO_ELEMENTS_EXPORT(), FIELD_IO_ROUTINES::FIELD_IO_EXPORT_ELEMENTS_INTO_(), FIELD_IO_ROUTINES::FIELD_IO_EXPORT_NODAL_GROUP_HEADER_FORTRAN(), FIELD_IO_ROUTINES::FIELD_IO_EXPORT_NODES_INTO_LOC(), FIELD_IO_ROUTINES::FIELD_IO_FIELD_INFO(), FIELD_IO_ROUTINES::FIELD_IO_FILEDS_GROUP_INFO_GET(), FIELD_IO_ROUTINES::FIELD_IO_FILEDS_IMPORT(), FIELD_IO_ROUTINES::FIELD_IO_FILL_BASIS_INFO(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_CLOSE(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_OPEN(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_DP(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_INTG(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_STRING(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_REWIND(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_DP(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_INTG(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_STRING(), FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH(), FIELD_IO_ROUTINES::FIELD_IO_LABEL_DERIVATIVE_INFO_GET(), FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET(), FIELD_IO_ROUTINES::FIELD_IO_MULTI_FILES_INFO_GET(), FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_ATTACH_LOCAL_PROCESS(), FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_FINALIZE(), FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_INITIALISE(), FIELD_IO_ROUTINES::FIELD_IO_NODES_EXPORT(), FIELD_IO_ROUTINES::FIELD_IO_TRANSLATE_LABEL_INTO_INTERPOLATION_TYPE(), FIELD_ROUTINES::FIELD_VARIABLE_COMPONENT_FINALISE(), FIELD_ROUTINES::FIELD_VARIABLE_COMPONENT_INITIALISE(), FIELD_ROUTINES::FIELD_VARIABLES_FINALISE(), FIELD_ROUTINES::FIELD_VARIABLES_INITIALISE(), FIELD_ROUTINES::FIELDS_FINALISE(), FIELD_ROUTINES::FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SETUP(), FIELD_ROUTINES::FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SUBTYPE_SET(), FIELD_ROUTINES::FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITYFINITE_ELEMENT_JACOBIAN_EVALUATE(), FIELD_ROUTINES::FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITYFINITE_ELEMENT_RESIDUAL_EVALUATE(), FIELD_ROUTINES::FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_GAUSS_CAUCHY_TENSOR(), FIELD_ROUTINES::FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_GAUSS_DEFORMATION_GRADEINT_TENSOR(), FIELD_ROUTINES::FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_GAUSS_DFDZ(), FIELD_ROUTINES::FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_GAUSS_DXDNU(), FIELD_ROUTINES::FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_PROBLEM_SETUP(), FIELD_ROUTINES::FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_PROBLEM_SUBTYPE_SET(), FIELD_ROUTINES::GENERATED_MESH_ROUTINES::GENERATED_MESH_MESH_BASIS_GET(), FIELD_ROUTINES::GENERATED_MESH_ROUTINES::GENERATED_MESH_BASIS_SET_NUMBER(), FIELD_ROUTINES::GENERATED_MESH_ROUTINES::GENERATED_MESH_BASIS_SET_PTR(), FIELD_ROUTINES::GENERATED_MESH_ROUTINES::GENERATED_MESH_CREATE_FINISH(), FIELD_ROUTINES::GENERATED_MESH_ROUTINES::GENERATED_MESH_CREATE_START(), FIELD_ROUTINES::GENERATED_MESH_ROUTINES::GENERATED_MESH_DESTROY_NUMBER(), FIELD_ROUTINES::GENERATED_MESH_ROUTINES::GENERATED_MESH_DESTROY_PTR(), FIELD_ROUTINES::GENERATED_MESH_ROUTINES::GENERATED_MESH_EXTENT_GET(),

```

GENERATED_MESH_ROUTINES::GENERATED_MESH_EXTENT_SET_NUMBER(),
GENERATED_MESH_ROUTINES::GENERATED_MESH_EXTENT_SET_PTR(),
GENERATED_MESH_ROUTINES::GENERATED_MESH_FINALISE(),           GENERATED_-
MESH_ROUTINES::GENERATED_MESH_INITIALISE(),                  GENERATED_MESH_-
ROUTINES::GENERATED_MESH_NUMBER_OF_ELEMENTS_GET(),          GENERATED_-
MESH_ROUTINES::GENERATED_MESH_NUMBER_OF_ELEMENTS_SET_NUMBER(),
GENERATED_MESH_ROUTINES::GENERATED_MESH_NUMBER_OF_ELEMENTS_-
SET_PTR(),          GENERATED_MESH_ROUTINES::GENERATED_MESH_ORIGIN_-
GET(),             GENERATED_MESH_ROUTINES::GENERATED_MESH_ORIGIN_SET_-
NUMBER(),          GENERATED_MESH_ROUTINES::GENERATED_MESH_ORIGIN_SET_-
PTR(),             GENERATED_MESH_ROUTINES::GENERATED_MESH_REGULAR_CREATE_-
FINISH(),          GENERATED_MESH_ROUTINES::GENERATED_MESH_REGULAR_-
FINALISE(),         GENERATED_MESH_ROUTINES::GENERATED_MESH_REGULAR_-
INITIALISE(),       GENERATED_MESH_ROUTINES::GENERATED_MESH_TYPE_GET(),
GENERATED_MESH_ROUTINES::GENERATED_MESH_TYPE_SET_NUMBER(),
GENERATED_MESH_ROUTINES::GENERATED_MESH_TYPE_SET_PTR(),        GENERATED_-
MESH_ROUTINES::GENERATED_MESH_USER_NUMBER_FIND(),
GENERATED_MESH_ROUTINES::GENERATED_MESHES_FINALISE(),          GENERATED_MESH_-
ROUTINES::GENERATED_MESHES_INITIALISE(),                     SORTING::HEAP_SORT_DP(),
SORTING::HEAP_SORT_INTG(),      SORTING::HEAP_SORT_SP(),     BINARY_FILE::INQUIRE_-
EOF_BINARY_FILE(), MATHS::INVERT_FULL_DP(), MATHS::INVERT_FULL_SP(), LAPLACE_-
EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_CALCULATE(),   LAPLACE_-
EQUATIONS_ROUTINES::LAPLACE_EQUATION_ANALYTIC_PARAMETER_SET_UPDATE(),
LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUIVATIONS_SET_SETUP(),
LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUIVATIONS_SET_STANDARD_-
SETUP(),          LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUIVATIONS_SET_-
SUBTYPE_SET(),      LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATIONFINITE_-
ELEMENT_CALCULATE(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-
FIXED_CONDITIONS_(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-
INTE(),            LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SETUP(),
LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(),
LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SUBTYPE_SET(),
LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUIVATIONS_SET_SETUP(),
LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUIVATIONS_SET_SUBTYPE_-
SET(),            LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITYFINITE_ELEMENT_-
CALCULATE(),       LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_-
SETUP(),          LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SUBTYPE_-
SET(),            LISTS::LIST_CREATE_FINISH(),    LISTS::LIST_CREATE_START(),    LISTS::LIST_-
DATA_TYPE_SET(),   LISTS::LIST_DESTROY(),      LISTS::LIST_DETACH_AND_DESTROY_DP(),
LISTS::LIST_DETACH_AND_DESTROY_INTG(),   LISTS::LIST_DETACH_AND_DESTROY_-
SP(),             LISTS::LIST_FINALISE(),     LISTS::LIST_INITIAL_SIZE_SET(),
LISTS::LIST_INITIALISE(),   LISTS::LIST_ITEM_ADD_DP1(),  LISTS::LIST_ITEM_ADD_-
SP1(),            LISTS::LIST_ITEM_DELETE(),  LISTS::LIST_ITEM_IN_LIST_DP1(),  LISTS::LIST_-
ITEM_IN_LIST_INTG1(), LISTS::LIST_ITEM_IN_LIST_SP1(),  LISTS::LIST_NUMBER_OF_-
ITEMS_GET(),      LISTS::LIST_REMOVE_DUPLICATES(), LISTS::LIST_SEARCH_DP_ARRAY(),
LISTS::LIST_SEARCH_INTG_ARRAY(),      LISTS::LIST_SEARCH_LINEAR_DP_ARRAY(),
LISTS::LIST_SEARCH_LINEAR_INTG_ARRAY(), LISTS::LIST_SEARCH_LINEAR_SP_-
ARRAY(),          LISTS::LIST_SEARCH_SP_ARRAY(),  LISTS::LIST_SORT_BUBBLE_DP_ARRAY(),
LISTS::LIST_SORT_BUBBLE_INTG_ARRAY(),  LISTS::LIST_SORT_BUBBLE_SP_ARRAY(),
LISTS::LIST_SORT_DP_ARRAY(),          LISTS::LIST_SORT_HEAP_DP_ARRAY(),  LISTS::LIST_-
SORT_HEAP_INTG_ARRAY(),   LISTS::LIST_SORT_HEAP_SP_ARRAY(),  LISTS::LIST_-
SORT_INTG_ARRAY(),      LISTS::LIST_SORT_SHELL_DP_ARRAY(),   LISTS::LIST_SORT_-
SHELL_INTG_ARRAY(),    LISTS::LIST_SORT_SHELL_SP_ARRAY(),  LISTS::LIST_SORT_-
SP_ARRAY(),         STRINGS::LIST_TO_CHARACTER_C(),   STRINGS::LIST_TO_CHARACTER_-

```

```

DP(),      STRINGS::LIST_TO_CHARACTER_INTG(),      STRINGS::LIST_TO_CHARACTER_-
L(),      STRINGS::LIST_TO_CHARACTER_LINTG(),      STRINGS::LIST_TO_CHARACTER_-
SP(),      STRINGS::LOGICAL_TO_CHARACTER(),      STRINGS::LOGICAL_TO_VSTRING(),
MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_DP(),      MATRIX_VECTOR::MATRIX_ALL_-
VALUES_SET_INTG(),      MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_L(),      MATRIX_-
VECTOR::MATRIX_ALL_VALUES_SET_SP(),      MATRIX_VECTOR::MATRIX_CREATE_FINISH(),
MATRIX_VECTOR::MATRIX_CREATE_START(),      MATRIX_VECTOR::MATRIX_DATA_GET_-
DP(),      MATRIX_VECTOR::MATRIX_DATA_GET_INTG(),      MATRIX_VECTOR::MATRIX_DATA_-
GET_L(),      MATRIX_VECTOR::MATRIX_DATA_GET_SP(),      MATRIX_VECTOR::MATRIX_DATA_-
TYPE_SET(),      MATRIX_VECTOR::MATRIX_DESTROY(),      MATRIX_VECTOR::MATRIX_-
DUPLICATE(),      MATRIX_VECTOR::MATRIX_FINALISE(),      MATRIX_VECTOR::MATRIX_-
INITIALISE(),      MATRIX_VECTOR::MATRIX_MAX_COLUMNS_PER_ROW_GET(),      MATRIX_-
VECTOR::MATRIX_MAX_SIZE_SET(),      MATRIX_VECTOR::MATRIX_NUMBER_NON_-
ZEROS_SET(),      MATRIX_VECTOR::MATRIX_OUTPUT(),      MATHS::MATRIX_PRODUCT_DP(),
MATHS::MATRIX_PRODUCT_SP(),      MATRIX_VECTOR::MATRIX_SIZE_SET(),      MATRIX_-
VECTOR::MATRIX_STORAGE_LOCATION_FIND(),      MATRIX_VECTOR::MATRIX_STORAGE_-
LOCATIONS_GET(),      MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_SET(),      MATRIX_-
VECTOR::MATRIX_STORAGE_TYPE_GET(),      MATRIX_VECTOR::MATRIX_STORAGE_-
TYPE_SET(),      MATHS::MATRIX_TRANSPOSE_DP(),      MATHS::MATRIX_TRANSPOSE_-
SP(),      MATRIX_VECTOR::MATRIX_VALUES_ADD_DP(),      MATRIX_VECTOR::MATRIX_-
VALUES_ADD_DP1(),      MATRIX_VECTOR::MATRIX_VALUES_ADD_DP2(),      MATRIX_-
VECTOR::MATRIX_VALUES_ADD_INTG(),      MATRIX_VECTOR::MATRIX_VALUES_ADD_-
INTG1(),      MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG2(),      MATRIX_VECTOR::MATRIX_-
VALUES_ADD_L(),      MATRIX_VECTOR::MATRIX_VALUES_ADD_L1(),      MATRIX_-
VECTOR::MATRIX_VALUES_ADD_L2(),      MATRIX_VECTOR::MATRIX_VALUES_ADD_-
SP(),      MATRIX_VECTOR::MATRIX_VALUES_ADD_SP1(),      MATRIX_VECTOR::MATRIX_-
VALUES_ADD_SP2(),      MATRIX_VECTOR::MATRIX_VALUES_GET_DP(),      MATRIX_-
VECTOR::MATRIX_VALUES_GET_DP1(),      MATRIX_VECTOR::MATRIX_VALUES_GET_-
DP2(),      MATRIX_VECTOR::MATRIX_VALUES_GET_INTG(),      MATRIX_VECTOR::MATRIX_-
VALUES_GET_INTG1(),      MATRIX_VECTOR::MATRIX_VALUES_GET_INTG2(),      MATRIX_-
VECTOR::MATRIX_VALUES_GET_L(),      MATRIX_VECTOR::MATRIX_VALUES_GET_L1(),
MATRIX_VECTOR::MATRIX_VALUES_GET_L2(),      MATRIX_VECTOR::MATRIX_VALUES_-
GET_SP(),      MATRIX_VECTOR::MATRIX_VALUES_GET_SP1(),      MATRIX_VECTOR::MATRIX_-
VALUES_GET_SP2(),      MATRIX_VECTOR::MATRIX_VALUES_SET_DP(),      MATRIX_-
VECTOR::MATRIX_VALUES_SET_DP1(),      MATRIX_VECTOR::MATRIX_VALUES_SET_-
DP2(),      MATRIX_VECTOR::MATRIX_VALUES_SET_INTG(),      MATRIX_VECTOR::MATRIX_-
VALUES_SET_INTG1(),      MATRIX_VECTOR::MATRIX_VALUES_SET_INTG2(),      MATRIX_-
VECTOR::MATRIX_VALUES_SET_L(),      MATRIX_VECTOR::MATRIX_VALUES_SET_L1(),
MATRIX_VECTOR::MATRIX_VALUES_SET_L2(),      MATRIX_VECTOR::MATRIX_VALUES_-
SET_SP(),      MATRIX_VECTOR::MATRIX_VALUES_SET_SP1(),      MATRIX_VECTOR::MATRIX_-
VALUES_SET_SP2(),      MESH_ROUTINES::MESH_CREATE_FINISH(),      MESH_ROUTINES::MESH_-
CREATE_START(),      MESH_ROUTINES::MESH_DESTROY(),      MESH_ROUTINES::MESH_-
FINALISE(),      MESH_ROUTINES::MESH_INITIALISE(),      MESH_ROUTINES::MESH_NUMBER_-
OF_COMPONENTS_GET(),      MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_-
SET_NUMBER(),      MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET_PTR(),
MESH_ROUTINES::MESH_NUMBER_OF_ELEMENTS_GET(),      MESH_ROUTINES::MESH_-
NUMBER_OF_ELEMENTS_SET_NUMBER(),      MESH_ROUTINES::MESH_NUMBER_OF_-
ELEMENTS_SET_PTR(),      MESH_ROUTINES::MESH_TOPOLOGY_CALCULATE(),      MESH_-
ROUTINES::MESH_TOPOLOGY_DOFS_CALCULATE(),      MESH_ROUTINES::MESH_-
TOPOLOGY_DOFS_FINALISE(),      MESH_ROUTINES::MESH_TOPOLOGY_DOFS_INITIALISE(),
MESH_ROUTINES::MESH_TOPOLOGY_ELEMENT_FINALISE(),      MESH_ROUTINES::MESH_-
TOPOLOGY_ELEMENT_INITIALISE(),      MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_-
ADJACENT_ELEMENTS_CALCULATE(),      MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_-
ELEMENTS_BASIS_SET(),      MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_-

```

FINISH(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_START(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_DESTROY(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_BASIS_GET(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_BASIS_SET(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_NODES_GET(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_NODES_SET(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_FINALISE(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_INITIALISE(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_NUMBER_GET(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_NUMBER_SET(), MESH_ROUTINES::MESH_TOPOLOGY_FINALISE(), MESH_ROUTINES::MESH_TOPOLOGY_INITIALISE(), MESH_ROUTINES::MESH_TOPOLOGY_NODE_FINALISE(), MESH_ROUTINES::MESH_TOPOLOGY_NODE_INITIALISE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_FINALISE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_INITIALISE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_SURROUNDING_ELEMENTS_CALCULATE(), MESH_ROUTINES::MESH_USER_NUMBER_FIND(), MESH_ROUTINES::MESHS_FINALISE(), MESH_ROUTINES::MESHS_INITIALISE(), CMISS_MPI::MPI_ERROR_CHECK(), NODE_ROUTINES::NODE_CHECK_EXISTS(), NODE_ROUTINES::NODE_DESTROY(), NODE_ROUTINES::NODE_INITIAL_POSITION_SET(), NODE_ROUTINES::NODE_NUMBER_SET(), NODE_ROUTINES::NODES_CREATE_FINISH(), NODE_ROUTINES::NODES_CREATE_START(), NODE_ROUTINES::NODES_FINALISE(), NODE_ROUTINES::NODES_INITIALISE(), MATHS::NORMALISE_DP(), MATHS::NORMALISE_SP(), STRINGS::NUMBER_TO_CHARACTER_DP(), STRINGS::NUMBER_TO_CHARACTER_INTG(), STRINGS::NUMBER_TO_CHARACTER_SP(), STRINGS::NUMBER_TO_VSTRING_SP(), BINARY_FILE::OPEN_BINARY_FILE(), BINARY_FILE::OPEN_CMISS_BINARY_FILE(), CMISS_PARMETIS::PARMETIS_PARTKWAY(), CMISS_PARMETIS::PARMETIS_PARTMESHKWAY(), CMISS_PETSC::PETSC_ERRORHANDLING_SET_OFF(), CMISS_PETSC::PETSC_ERRORHANDLING_SET_ON(), CMISS_PETSC::PETSC_FINALIZE(), CMISS_PETSC::PETSC_INITIALIZE(), CMISS_PETSC::PETSC_ISCOLORINGDESTROY(), CMISS_PETSC::PETSC_ISCOLORINGFINALISE(), CMISS_PETSC::PETSC_ISCOLORINGINITIALISE(), CMISS_PETSC::PETSC_ISDESTROY(), CMISS_PETSC::PETSC_ISFINALISE(), CMISS_PETSC::PETSC_ISINITIALISE(), CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGAPPLY(), CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGAPPLYIS(), CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGCREATE(), CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGDESTROY(), CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGFINALISE(), CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGINITIALISE(), CMISS_PETSC::PETSC_KSPCREATE(), CMISS_PETSC::PETSC_KSPDESTROY(), CMISS_PETSC::PETSC_KSPFINALISE(), CMISS_PETSC::PETSC_KSPGETCONVERGEDREASON(), CMISS_PETSC::PETSC_KSPGETITERATIONNUMBER(), CMISS_PETSC::PETSC_KSPGETPC(), CMISS_PETSC::PETSC_KSPGETRESIDUALNORM(), CMISS_PETSC::PETSC_KSPINITIALISE(), CMISS_PETSC::PETSC_KSPSETFROMOPTIONS(), CMISS_PETSC::PETSC_KSPSETOPERATORS(), CMISS_PETSC::PETSC_KSPSETTOLERANCES(), CMISS_PETSC::PETSC_KSPSETTYPE(), CMISS_PETSC::PETSC_KSPSETUP(), CMISS_PETSC::PETSC_KSPSOLVE(), CMISS_PETSC::PETSC_LOGPRINTSUMMARY(), CMISS_PETSC::PETSC_MATASEMBLYBEGIN(), CMISS_PETSC::PETSC_MATASEMBLYEND(), CMISS_PETSC::PETSC_MATCREATE(), CMISS_PETSC::PETSC_MATCREATEMPIAIJ(), CMISS_PETSC::PETSC_MATCREATETEMPDENSE(), CMISS_PETSC::PETSC_MATCREATESEQDENSE(), CMISS_PETSC::PETSC_MATDESTROY(), CMISS_PETSC::PETSC_MATFDCOLORINGCREATE(), CMISS_PETSC::PETSC_MATFDCOLORINGDESTROY(), CMISS_PETSC::PETSC_MATFDCOLORINGFINALISE(), CMISS_PETSC::PETSC_MATFDCOLORINGINITIALISE(), CMISS_PETSC::PETSC_MATFDCOLORINGSETFROMOPTIONS(), CMISS_PETSC::PETSC_MATFINALISE(),

```

CMISS_PETSC::PETSC_MATGETARRAY(),      CMISS_PETSC::PETSC_MATGETCOLORING(),
CMISS_PETSC::PETSC_MATGETOWNERSHIPRANGE(),   CMISS_PETSC::PETSC_-
MATGETVALUES(),    CMISS_PETSC::PETSC_MATINITIALISE(),   CMISS_PETSC::PETSC_-
MATRESTOREARRAY(),   CMISS_PETSC::PETSC_MATSETLOCALTOGLOBALMAPPING(),
CMISS_PETSC::PETSC_MATSETOPTION(),  CMISS_PETSC::PETSC_MATSETSIZES(),  CMISS_-
PETSC::PETSC_MATSETVALUE(),  CMISS_PETSC::PETSC_MATSETVALUELOCAL(),  CMISS_-
PETSC::PETSC_MATSETVALUES(),       CMISS_PETSC::PETSC_MATSETVALUESLOCAL(),
CMISS_PETSC::PETSC_MATVIEW(),        CMISS_PETSC::PETSC_MATZEROENTRIES(),
CMISS_PETSC::PETSC_PCFINALISE(),     CMISS_PETSC::PETSC_PCINITIALISE(),
CMISS_PETSC::PETSC_PCSETTYPE(),      CMISS_PETSC::PETSC_SNESCREATE(),
CMISS_PETSC::PETSC_SNESDESTROY(),    CMISS_PETSC::PETSC_SNESFINALISE(),
CMISS_PETSC::PETSC_SNESGETCONVERGEDREASON(),   CMISS_PETSC::PETSC_-
SNESGETFUNCTIONNORM(),    CMISS_PETSC::PETSC_SNESGETITERATIONNUMBER(),
CMISS_PETSC::PETSC_SNESINITIALISE(),   CMISS_PETSC::PETSC_SNESLINESEARCHSET(),
CMISS_PETSC::PETSC_SNESLINESEARCHSETPARAMS(),   CMISS_PETSC::PETSC_-
SNESSETFROMOPTIONS(),    CMISS_PETSC::PETSC_SNESSETFUNCTION(),   CMISS_-
PETSC::PETSC_SNESSETJACOBIAN_MATFDCOLORING(),   CMISS_PETSC::PETSC_-
SNESSETJACOBIAN_SOLVER(),       CMISS_PETSC::PETSC_SNESSETTOLERANCES(),
CMISS_PETSC::PETSC_SNESSETTRUSTREGIONTOLERANCE(),   CMISS_PETSC::PETSC_-
SNESSETTYPE(),    CMISS_PETSC::PETSC_SNESSOLVE(),   CMISS_PETSC::PETSC_-
VECASSEMBLYBEGIN(),    CMISS_PETSC::PETSC_VECASSEMBLYEND(),   CMISS_-
PETSC::PETSC_VECCREATE(),    CMISS_PETSC::PETSC_VECCREATEGHOST(),   CMISS_-
PETSC::PETSC_VECCREATEGHOSTWITHARRAY(), CMISS_PETSC::PETSC_VECCREATEMPI(),
CMISS_PETSC::PETSC_VECCREATEMPIWITHARRAY(),   CMISS_PETSC::PETSC_-
VECCREATESEQ(),    CMISS_PETSC::PETSC_VECCREATESEQWITHARRAY(),   CMISS_-
PETSC::PETSC_VECDESTROY(),    CMISS_PETSC::PETSC_VECDUPLICATE(),   CMISS_-
PETSC::PETSC_VECFINALISE(),   CMISS_PETSC::PETSC_VECGETARRAY(),   CMISS_-
PETSC::PETSC_VECGETARRAYF90(),   CMISS_PETSC::PETSC_VECGETLOCALSIZE(),
CMISS_PETSC::PETSC_VECGETOWNERSHIPRANGE(),   CMISS_PETSC::PETSC_-
VECGETSIZE(),    CMISS_PETSC::PETSC_VECGETVALUES(),   CMISS_PETSC::PETSC_-
VECHOSTGETLOCALFORM(),  CMISS_PETSC::PETSC_VECHOSTRESTORELOCALFORM(),
CMISS_PETSC::PETSC_VECHOSTUPDATEBEGIN(),   CMISS_PETSC::PETSC_-
VECHOSTUPDATEEND(),   CMISS_PETSC::PETSC_VECINITIALISE(),  CMISS_PETSC::PETSC_-
VCRESTOREARRAY(),    CMISS_PETSC::PETSC_VCRESTOREARRAYF90(),   CMISS_-
PETSC::PETSC_VECSET(),    CMISS_PETSC::PETSC_VECSETFROMOPTIONS(),   CMISS_-
PETSC::PETSC_VECSETLOCALTOGLOBALMAPPING(),   CMISS_PETSC::PETSC_-
VECSETSIZES(),    CMISS_PETSC::PETSC_VECSETVALUES(),   CMISS_PETSC::PETSC_-
VECSETVALUESLOCAL(),   CMISS_PETSC::PETSC_VECVIEW(),   PROBLEM_-
ROUTINES::PROBLEM_CONTROL_CREATE_FINISH(),   PROBLEM_ROUTINES::PROBLEM_-
CONTROL_CREATE_START(),   PROBLEM_ROUTINES::PROBLEM_CONTROL_-
DESTROY(),    PROBLEM_ROUTINES::PROBLEM_CONTROL_FINALISE(),   PROBLEM_-
ROUTINES::PROBLEM_CONTROL_INITIALISE(),   PROBLEM_ROUTINES::PROBLEM_-
CREATE_FINISH(),   PROBLEM_ROUTINES::PROBLEM_CREATE_START(),   PROBLEM_-
ROUTINES::PROBLEM_DESTROY_NUMBER(),   PROBLEM_ROUTINES::PROBLEM_-
DESTROY_PTR(),   PROBLEM_ROUTINES::PROBLEM_FINALISE(),   PROBLEM_-
ROUTINES::PROBLEM_INITIALISE(),   PROBLEM_ROUTINES::PROBLEM_SETUP(),
PROBLEM_ROUTINES::PROBLEM SOLUTION_EQUATIONS_SET_ADD(),   PROBLEM_-
ROUTINES::PROBLEM SOLUTION_FINALISE(),   PROBLEM_ROUTINES::PROBLEM_-
SOLUTION_INITIALISE(),   PROBLEM_ROUTINES::PROBLEM SOLUTION_JACOBIAN_-
EVALUATE(),    PROBLEM_ROUTINES::PROBLEM SOLUTION_RESIDUAL_EVALUATE(),
PROBLEM_ROUTINES::PROBLEM SOLUTION_SOLVE(),   PROBLEM_ROUTINES::PROBLEM_-
SOLUTIONS_CREATE_FINISH(),  PROBLEM_ROUTINES::PROBLEM SOLUTIONS_CREATE_-
START(),    PROBLEM_ROUTINES::PROBLEM SOLUTIONS_FINALISE(),   PROBLEM_-
ROUTINES::PROBLEM SOLUTIONS_INITIALISE(),   PROBLEM_ROUTINES::PROBLEM_-

```

SOLVE(), PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_FINISH(), PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_START(), PROBLEM_ROUTINES::PROBLEM_SOLVER_DESTROY(), PROBLEM_ROUTINES::PROBLEM_SOLVER_GET(), PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_GET_NUMBER(), PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_GET_PTR(), PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_NUMBER(), PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_PTR(), PROBLEM_ROUTINES::PROBLEM_USER_NUMBER_FIND(), PROBLEMS_FINALISE(), PROBLEM_ROUTINES::PROBLEMS_INITIALISE(), BINARY_FILE::READ_BINARY_FILE_CHARACTER(), BINARY_FILE::READ_BINARY_FILE_DP(), BINARY_FILE::READ_BINARY_FILE_DP1(), BINARY_FILE::READ_BINARY_FILE_DPC(), BINARY_FILE::READ_BINARY_FILE_DPC1(), BINARY_FILE::READ_BINARY_FILE_INTG(), BINARY_FILE::READ_BINARY_FILE_INTG1(), BINARY_FILE::READ_BINARY_FILE_LINTG(), BINARY_FILE::READ_BINARY_FILE_LINTG1(), BINARY_FILE::READ_BINARY_FILE_LOGICAL(), BINARY_FILE::READ_BINARY_FILE_LOGICAL1(), BINARY_FILE::READ_BINARY_FILE_SINTG(), BINARY_FILE::READ_BINARY_FILE_SINTG1(), BINARY_FILE::READ_BINARY_FILE_SP(), BINARY_FILE::READ_BINARY_FILE_SP1(), BINARY_FILE::READ_BINARY_FILE_SPC(), BINARY_FILE::READ_BINARY_FILE_SPC1(), BINARY_FILE::READ_BINARY_TAG_HEADER(), REGION_ROUTINES::REGION_COORDINATE_SYSTEM_GET(), REGION_ROUTINES::REGION_COORDINATE_SYSTEM_SET_NUMBER(), REGION_ROUTINES::REGION_CREATE_FINISH(), REGION_ROUTINES::REGION_CREATE_START(), REGION_ROUTINES::REGION_DESTROY(), REGION_ROUTINES::REGION_LABEL_GET(), REGION_ROUTINES::REGION_LABEL_SET_NUMBER(), REGION_ROUTINES::REGION_LABEL_SET_PTR(), REGION_ROUTINES::REGION_SUB_REGION_CREATE_FINISH(), REGION_ROUTINES::REGION_SUB_REGION_CREATE_START(), REGION_ROUTINES::REGION_USER_NUMBER_FIND_PTR(), REGION_ROUTINES::REGIONS_FINALISE(), REGION_ROUTINES::REGIONS_INITIALISE(), BINARY_FILE::RESET_BINARY_NUMBER_TAGS(), BINARY_FILE::SET_BINARY_FILE(), SORTING::SHELL_SORT_DP(), SORTING::SHELL_SORT_INTG(), SORTING::SHELL_SORT_SP(), BINARY_FILE::SKIP_BINARY_FILE(), BINARY_FILE::SKIP_BINARY_TAGS(), BINARY_FILE::SKIP_CM_BINARY_HEADER(), MATHS::SOLVE_SMALL_LINEAR_SYSTEM_DP(), MATHS::SOLVE_SMALL_LINEAR_SYSTEM_SP(), SOLVER_ROUTINES::SOLVER_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_CREATE_START(), SOLVER_ROUTINES::SOLVER_DESTROY(), SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_FINALISE(), SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_INITIALISE(), SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_SOLVE(), SOLVER_ROUTINES::SOLVER_FINALISE(), SOLVER_ROUTINES::SOLVER_INITIALISE(), SOLVER_ROUTINES::SOLVER_LIBRARY_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_FINALISE(), SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_INITIALISE(), SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_SOLVE(), SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_TYPE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_FINALISE(), SOLVER_ROUTINES::SOLVER_LINEAR_INITIALISE(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_ABSOLUTE_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_DIVergence_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_FINALISE(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_INITIALISE(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_MAXIMUM_ITERATIONS_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_RELATIVE_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_SOLVE(), SOLVER_ROUTINES::SOLVER_LINEAR_TYPE_SET(), SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE(),

```

SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_FINISH(),           SOLVER_-
MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_START(),           SOLVER_MATRICES_-
ROUTINES::SOLVER_MATRICES_DESTROY(),           SOLVER_MATRICES_ROUTINES::SOLVER_-
MATRICES_FINALISE(),           SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_INITIALISE(), 
SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_LIBRARY_TYPE_GET(),           SOLVER_-
MATRICES_ROUTINES::SOLVER_MATRICES_LIBRARY_TYPE_SET(),           SOLVER_MATRICES_-
ROUTINES::SOLVER_MATRICES_OUTPUT(),           SOLVER_MATRICES_ROUTINES::SOLVER_-
MATRICES_STORAGE_TYPE_GET(),           SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_-
STORAGE_TYPE_SET(),           SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_FINALISE(), 
SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_FORM(),           SOLVER_MATRICES_-
ROUTINES::SOLVER_MATRIX_INITIALISE(),           SOLVER_MATRICES_ROUTINES::SOLVER_-
MATRIX_STRUCTURE_CALCULATE(),           SOLVER_ROUTINES::SOLVER_NONLINEAR_-
ABSOLUTE_TOLERANCE_SET(),           SOLVER_ROUTINES::SOLVER_NONLINEAR_-
CREATE_FINISH(),           SOLVER_ROUTINES::SOLVER_NONLINEAR_FINALISE(),           SOLVER_-
ROUTINES::SOLVER_NONLINEAR_INITIALISE(),           SOLVER_ROUTINES::SOLVER_-
NONLINEAR_JACOBIAN_EVALUATE(),           SOLVER_ROUTINES::SOLVER_NONLINEAR_-
LINESEARCH_ALPHA_SET(),           SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_-
CREATE_FINISH(),           SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_-
FINALISE(),           SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_INITIALISE(), 
SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_MAXSTEP_SET(),           SOLVER_-
ROUTINES::SOLVER_NONLINEAR_LINESEARCH_SOLVE(),           SOLVER_ROUTINES::SOLVER_-
NONLINEAR_LINESEARCH_STEPTOL_SET(),           SOLVER_ROUTINES::SOLVER_NONLINEAR_-
LINESEARCH_TYPE_SET(),           SOLVER_ROUTINES::SOLVER_NONLINEAR_MAXIMUM_-
FUNCTION_EVALUATIONS_SET(),           SOLVER_ROUTINES::SOLVER_NONLINEAR_-
MAXIMUM_ITERATIONS_SET(),           SOLVER_ROUTINES::SOLVER_NONLINEAR_-
RELATIVE_TOLERANCE_SET(),           SOLVER_ROUTINES::SOLVER_NONLINEAR_RESIDUAL_-
EVALUATE(),           SOLVER_ROUTINES::SOLVER_NONLINEAR SOLUTION_TOLERANCE_SET(), 
SOLVER_ROUTINES::SOLVER_NONLINEAR_SOLVE(),           SOLVER_ROUTINES::SOLVER_-
NONLINEAR_TRUSTREGION_CREATE_FINISH(),           SOLVER_ROUTINES::SOLVER_-
NONLINEAR_TRUSTREGION_DELTA0_SET(),           SOLVER_ROUTINES::SOLVER_NONLINEAR_-
TRUSTREGION_FINALISE(),           SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_-
INITIALISE(),           SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_SOLVE(), 
SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_TOLERANCE_SET(),           SOLVER_-
ROUTINES::SOLVER_NONLINEAR_TYPE_SET(),           SOLVER_ROUTINES::SOLVER_OUTPUT_-
TYPE_SET(),           SOLVER_ROUTINES::SOLVER_SOLVE(),           SOLVER_ROUTINES::SOLVER_-
SPARSITY_TYPE_SET(),           SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_-
CREATE_FINISH(),           SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_FINALISE(), 
SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_INITIALISE(),           SOLVER_-
ROUTINES::SOLVER_TIME_INTEGRATION_SOLVE(),           SOLVER_ROUTINES::SOLVER_-
VARIABLES_UPDATE(),           STRINGS::STRING_TO_DOUBLE_C(),           STRINGS::STRING_-
TO_DOUBLE_VS(),           STRINGS::STRING_TO_INTEGER_C(),           STRINGS::STRING_TO_-
INTEGER_VS(),           STRINGS::STRING_TO_LOGICAL_C(),           STRINGS::STRING_TO_LOGICAL_-
VS(),           STRINGS::STRING_TO_LONG_INTEGER_C(),           STRINGS::STRING_TO_LONG_-
INTEGER_VS(),           FIELD_IO_ROUTINES::STRING_TO_MUTI_INTEGERS_VS(),           FIELD_-
IO_ROUTINES::STRING_TO_MUTI_REALS_VS(),           STRINGS::STRING_TO_SINGLE_C(), 
STRINGS::STRING_TO_SINGLE_VS(),           TREES::TREE_CREATE_FINISH(),           TREES::TREE_-
CREATE_START(),           TREES::TREE_DESTROY(),           TREES::TREE_DETACH_AND_DESTROY(), 
TREES::TREE_DETACH_IN_ORDER(),           TREES::TREE_FINALISE(),           TREES::TREE_INITIALISE(), 
TREES::TREE_INSERT_TYPE_SET(),           TREES::TREE_ITEM_DELETE(),           TREES::TREE_-
ITEM_INSERT(),           TREES::TREE_NODE_FINALISE(),           TREES::TREE_NODE_INITIALISE(), 
TREES::TREE_NODE_KEY_GET(),           TREES::TREE_NODE_VALUE_GET(),           TREES::TREE_-
NODE_VALUE_SET(),           TREES::TREE_OUTPUT(),           TREES::TREE_OUTPUT_IN_ORDER(), 
TREES::TREE_PREDECESSOR(),           TREES::TREE_SEARCH(),           TREES::TREE_SUCCESSOR(), 
MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_DP(),           MATRIX_VECTOR::VECTOR_ALL_-

```

VALUES_SET_INTG(), MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_L(), MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_SP(), MATRIX_VECTOR::VECTOR_CREATE_FINISH(), MATRIX_VECTOR::VECTOR_CREATE_START(), MATRIX_VECTOR::VECTOR_DATA_GET_DP(), MATRIX_VECTOR::VECTOR_DATA_GET_INTG(), MATRIX_VECTOR::VECTOR_DATA_GET_L(), MATRIX_VECTOR::VECTOR_DATA_GET_SP(), MATRIX_VECTOR::VECTOR_DATA_TYPE_SET(), MATRIX_VECTOR::VECTOR_DESTROY(), MATRIX_VECTOR::VECTOR_DUPLICATE(), MATRIX_VECTOR::VECTOR_FINALISE(), MATRIX_VECTOR::VECTOR_INITIALISE(), MATRIX_VECTOR::VECTOR_SIZE_SET(), MATRIX_VECTOR::VECTOR_VALUES_GET_DP(), MATRIX_VECTOR::VECTOR_VALUES_GET_DP1(), MATRIX_VECTOR::VECTOR_VALUES_GET_INTG(), MATRIX_VECTOR::VECTOR_VALUES_GET_INTG1(), MATRIX_VECTOR::VECTOR_VALUES_GET_L(), MATRIX_VECTOR::VECTOR_VALUES_GET_L10(), MATRIX_VECTOR::VECTOR_VALUES_GET_SP(), MATRIX_VECTOR::VECTOR_VALUES_GET_SP1(), MATRIX_VECTOR::VECTOR_VALUES_SET_DP(), MATRIX_VECTOR::VECTOR_VALUES_SET_DP1(), MATRIX_VECTOR::VECTOR_VALUES_SET_INTG(), MATRIX_VECTOR::VECTOR_VALUES_SET_INTG1(), MATRIX_VECTOR::VECTOR_VALUES_SET_L(), MATRIX_VECTOR::VECTOR_VALUES_SET_L10(), MATRIX_VECTOR::VECTOR_VALUES_SET_SP(), MATRIX_VECTOR::VECTOR_VALUES_SET_SP1(), BINARY_FILE::WRITE_BINARY_FILE_CHARACTER(), BINARY_FILE::WRITE_BINARY_FILE_DP(), BINARY_FILE::WRITE_BINARY_FILE_DP1(), BINARY_FILE::WRITE_BINARY_FILE_DPC(), BINARY_FILE::WRITE_BINARY_FILE_DPC1(), BINARY_FILE::WRITE_BINARY_FILE_INTG(), BINARY_FILE::WRITE_BINARY_FILE_INTG1(), BINARY_FILE::WRITE_BINARY_FILE_LINTG(), BINARY_FILE::WRITE_BINARY_FILE_LINTG1(), BINARY_FILE::WRITE_BINARY_FILE_LOGICAL(), BINARY_FILE::WRITE_BINARY_FILE_LOGICAL10(), BINARY_FILE::WRITE_BINARY_FILE_SINTG(), BINARY_FILE::WRITE_BINARY_FILE_SINTG1(), BINARY_FILE::WRITE_BINARY_FILE_SP(), BINARY_FILE::WRITE_BINARY_FILE_SP1(), BINARY_FILE::WRITE_BINARY_FILE_SPC(), BINARY_FILE::WRITE_BINARY_FILE_SPC1(), and BINARY_FILE::WRITE_BINARY_TAG_HEADER().

6.2.2.8 subroutine BASE_ROUTINES::FLAG_ERROR_C (CHARACTER(LEN=*),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Sets the error string specified by a character string and flags an error.

Parameters:

STRING The error condition string

ERR The error code

ERROR The error string

Definition at line 470 of file base_routines.f90.

6.2.2.9 subroutine BASE_ROUTINES::FLAG_ERROR_VS (TYPE(VARYING_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Sets the error string specified by a varying string and flags an error.

Parameters:

STRING The error condition string

ERR The error code

ERROR The error string

Definition at line 491 of file base_routines.f90.

6.2.2.10 subroutine BASE_ROUTINES::FLAG_WARNING_C (CHARACTER(LEN=*),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Writes a warning message specified by a character string to the user.

Parameters:

STRING The warning string

ERR The error code

ERROR The error string

Definition at line 510 of file base_routines.f90.

6.2.2.11 subroutine BASE_ROUTINES::FLAG_WARNING_VS (TYPE(VARYING_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Writes a warning message specified by a varying string to the user.

Parameters:

STRING The warning string

ERR The error code

ERROR The error string

Definition at line 531 of file base_routines.f90.

6.2.2.12 subroutine BASE_ROUTINES::OUTPUT_SET_OFF (INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Sets writes file echo output off.

Parameters:

ERR The error code

ERROR The error string

Definition at line 830 of file base_routines.f90.

6.2.2.13 subroutine BASE_ROUTINES::OUTPUT_SET_ON (CHARACTER(LEN=*),intent(in) ECHO_FILENAME, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Sets writes file echo output on.

Parameters:

ECHO_FILENAME The filename of the file to echo output to
ERR The error code
ERROR The error string

Definition at line 858 of file base_routines.f90.

6.2.2.14 subroutine **BASE_ROUTINES::TIMING_SET_OFF** (INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING_STRING),intent(out) **ERROR**, *)

Sets timing off.

Parameters:

ERR The error code
ERROR The error string

Definition at line 890 of file base_routines.f90.

6.2.2.15 subroutine **BASE_ROUTINES::TIMING_SET_ON** (INTEGER(INTG),intent(in) **TIMING_TYPE**, LOGICAL,intent(in) **TIMING_SUMMARY_FLAG**, CHARACTER(LEN=*),intent(in) **TIMING_FILENAME**, CHARACTER(LEN=*),dimension(:),intent(in) **ROUTINE_LIST**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING_STRING),intent(out) **ERROR**, *)

Sets timing on.

Parameters:

TIMING_TYPE The type of timing to set on

See also:

[BASE_ROUTINES::TimingTypes](#)

TIMING_SUMMARY_FLAG .TRUE. if the timing information will be output with subsequent **TIMING_SUMMARY_OUTPUT** calls, .FALSE. if the timing information will be output every time the routine exits

TIMING_FILENAME If present the name of the file to output timing information to. If omitted the timing output is sent to the screen

ROUTINE_LIST The list of routines to set timing on in.

ERR The error code

ERROR The error string

Definition at line 936 of file base_routines.f90.

References **KINDS::_SP**.

6.2.2.16 subroutine **BASE_ROUTINES::TIMING_SUMMARY_OUTPUT** (INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING_STRING),intent(out) **ERROR**, *)

Outputs the timing summary.

Parameters:***ERR*** The error code***ERROR*** The error string

Definition at line 1059 of file base_routines.f90.

6.2.2.17 subroutine **BASE_ROUTINES::WRITE_STR** (INTEGER(INTG),intent(in) ***ID***, INTEGER(INTG),intent(out) ***ERR***, TYPE(VARYING_STRING),intent(out) ***ERROR***, *)

Writes the output string to a specified output stream.

Parameters:***ID*** The ID of the output stream. An ID of > 9 specifies file output**See also:**[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)***ERR*** The error code***ERROR*** The error string

Definition at line 1121 of file base_routines.f90.

References [MACHINE_CONSTANTS::MACHINE_OS](#).

Referenced by `INPUT_OUTPUT::WR()`, `INPUT_OUTPUT::WRITE_STRING_C()`, `INPUT_OUTPUT::WRITE_STRING_FMT()`, `INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_C()`, `INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_DP()`, `INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_INTG()`, `INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_L()`, `INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_SP()`, `INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_VS()`, `INPUT_OUTPUT::WRITE_STRING_IDX()`, `INPUT_OUTPUT::WRITE_STRING_MATRIX_DP()`, `INPUT_OUTPUT::WRITE_STRING_MATRIX_INTG()`, `INPUT_OUTPUT::WRITE_STRING_MATRIX_L()`, `INPUT_OUTPUT::WRITE_STRING_MATRIX_SP()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_C()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_DP()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_L()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_SP()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_VS()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_C()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_DP()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_INTG()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_L()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_SP()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_VS()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_C()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_DP()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_INTG()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_L()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_SP()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_VS()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_C()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_DP()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_INTG()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_L()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_SP()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_VS()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_C()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_DP()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_INTG()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_L()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_SP()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_VS()`, `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS()`.

`OUTPUT::WRITE_STRING_TWO_VALUE_VS_C(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_DP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_INTG(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_VS(), INPUT_OUTPUT::WRITE_STRING_VALUE_C(), INPUT_OUTPUT::WRITE_STRING_VALUE_DP(), INPUT_OUTPUT::WRITE_STRING_VALUE_INTG(), INPUT_OUTPUT::WRITE_STRING_VALUE_L(), INPUT_OUTPUT::WRITE_STRING_VALUE_LINTG(), INPUT_OUTPUT::WRITE_STRING_VALUE_SP(), INPUT_OUTPUT::WRITE_STRING_VALUE_VS(), and INPUT_OUTPUT::WRITE_STRING_VS()`.

6.2.3 Variable Documentation

6.2.3.1 LOGICAL,save BASE_ROUTINES::DIAG_ALL_SUBROUTINES

.TRUE. if diagnostic output is required in all routines

Definition at line 156 of file base_routines.f90.

6.2.3.2 LOGICAL,save BASE_ROUTINES::DIAG_FILE_OPEN

.TRUE. if the diagnostic output file is open

Definition at line 158 of file base_routines.f90.

6.2.3.3 LOGICAL,save BASE_ROUTINES::DIAG_FROM_SUBROUTINE

.TRUE. if diagnostic output is required from a particular routine

Definition at line 157 of file base_routines.f90.

6.2.3.4 LOGICAL,save BASE_ROUTINES::DIAG_OR_TIMING

.TRUE. if diagnostics or time is .TRUE.

Definition at line 159 of file base_routines.f90.

6.2.3.5 TYPE(ROUTINE_LIST_TYPE),save BASE_ROUTINES::DIAG_ROUTINE_LIST

The list of routines for which diagnostic output is required.

Definition at line 167 of file base_routines.f90.

6.2.3.6 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS

.TRUE. if diagnostic output is required in any routines.

Definition at line 145 of file base_routines.f90.

6.2.3.7 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS1

.TRUE. if level 1 diagnostic output is active in the current routine

Definition at line 146 of file base_routines.f90.

Referenced by COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_INITIALISE(), COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_FINISH(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE(), MESH_ROUTINES::DECOMPOSITION_CREATE_FINISH(), MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE(), MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(), MESH_ROUTINES::DOMAIN_TOPOLOGY_INITIALISE_FROM_MESH(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_CALCULATE(), FIELD_ROUTINES::FIELD_CREATE_FINISH(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), MESH_ROUTINES::MESH_CREATE_FINISH(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_FINISH(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE(), NODE_ROUTINES::NODES_CREATE_FINISH(), PROBLEM_ROUTINES::PROBLEM_CREATE_FINISH(), REGION_ROUTINES::REGION_CREATE_FINISH(), REGION_ROUTINES::REGION_SUB_REGION_CREATE_FINISH(), and SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_CALCULATE().

6.2.3.8 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS2

.TRUE. if level 2 diagnostic output is active in the current routine

Definition at line 147 of file base_routines.f90.

Referenced by COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_INITIALISE(), MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(), and FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

6.2.3.9 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS3

.TRUE. if level 3 diagnostic output is active in the current routine

Definition at line 148 of file base_routines.f90.

Referenced by COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_TYPE_INITIALISE().

6.2.3.10 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS4

.TRUE. if level 4 diagnostic output is active in the current routine

Definition at line 149 of file base_routines.f90.

6.2.3.11 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS5

.TRUE. if level 5 diagnostic output is active in the current routine

Definition at line 150 of file base_routines.f90.

6.2.3.12 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS_LEVEL1

.TRUE. if the user has requested level 1 diagnostic output to be active

Definition at line 151 of file base_routines.f90.

6.2.3.13 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS_LEVEL2

.TRUE. if the user has requested level 2 diagnostic output to be active

Definition at line 152 of file base_routines.f90.

6.2.3.14 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS_LEVEL3

.TRUE. if the user has requested level 3 diagnostic output to be active

Definition at line 153 of file base_routines.f90.

6.2.3.15 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS_LEVEL4

.TRUE. if the user has requested level 4 diagnostic output to be active

Definition at line 154 of file base_routines.f90.

6.2.3.16 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS_LEVEL5

.TRUE. if the user has requested level 5 diagnostic output to be active

Definition at line 155 of file base_routines.f90.

6.2.3.17 LOGICAL,save BASE_ROUTINES::ECHO_OUTPUT

.TRUE. if all output is to be echoed to the echo file

Definition at line 160 of file base_routines.f90.

6.2.3.18 INTEGER(INTG),parameter BASE_ROUTINES::MAX_OUTPUT_LINES = 500

Maximum number of lines that can be output.

See also:

[BASE_ROUTINES::WRITE_STR](#)

Definition at line 57 of file base_routines.f90.

6.2.3.19 CHARACTER(LEN=MAXSTRLEN),save BASE_ROUTINES::OP_STRING

The array of lines to output.

Definition at line 166 of file base_routines.f90.

Referenced by `BINARY_FILE::OPEN_CMISS_BINARY_FILE()`, `INPUT_OUTPUT::WR()`, `INPUT_OUTPUT::WRITE_STRING_C()`, `INPUT_OUTPUT::WRITE_STRING_FMT()`, `INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_C()`, `INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_DPO()`, `INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_INTG()`, `INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_REAL()`.

```

OUTPUT::WRITE_STRING_FMT_VALUE_L(),           INPUT_OUTPUT::WRITE_STRING_FMT_-
VALUE_LINTG(),      INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_SP(),      INPUT_-
OUTPUT::WRITE_STRING_FMT_VALUE_VS(),      INPUT_OUTPUT::WRITE_STRING_IDX(),
INPUT_OUTPUT::WRITE_STRING_MATRIX_DP(),      INPUT_OUTPUT::WRITE_STRING_-
MATRIX_INTG(),      INPUT_OUTPUT::WRITE_STRING_MATRIX_L(),      INPUT_OUTPUT::WRITE_-
STRING_MATRIX_LINTG(),      INPUT_OUTPUT::WRITE_STRING_MATRIX_SP(),      INPUT_-
OUTPUT::WRITE_STRING_TWO_VALUE_C_C(),      INPUT_OUTPUT::WRITE_STRING_TWO_-
VALUE_C_DP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_INTG(),      INPUT_-
OUTPUT::WRITE_STRING_TWO_VALUE_C_L(),      INPUT_OUTPUT::WRITE_STRING_-
TWO_VALUE_C_SP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_VS(),      INPUT_-
OUTPUT::WRITE_STRING_TWO_VALUE_DP_C(),      INPUT_OUTPUT::WRITE_STRING_TWO_-
VALUE_DP_DP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_INTG(),      INPUT_-
OUTPUT::WRITE_STRING_TWO_VALUE_DP_L(),      INPUT_OUTPUT::WRITE_STRING_-
TWO_VALUE_DP_SP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_VS(),      INPUT_-
OUTPUT::WRITE_STRING_TWO_VALUE_INTG_C(),      INPUT_OUTPUT::WRITE_STRING_-
TWO_VALUE_INTG_DP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_INTG(),
INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_L(),      INPUT_OUTPUT::WRITE_-
STRING_TWO_VALUE_INTG_SP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_-
VS(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_C(),      INPUT_OUTPUT::WRITE_-
STRING_TWO_VALUE_L_DP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_INTG(),
INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_L(),      INPUT_OUTPUT::WRITE_STRING_-
TWO_VALUE_L_SP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_VS(),      INPUT_-
OUTPUT::WRITE_STRING_TWO_VALUE_SP_C(),      INPUT_OUTPUT::WRITE_STRING_TWO_-
VALUE_SP_DP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_INTG(),      INPUT_-
OUTPUT::WRITE_STRING_TWO_VALUE_SP_L(),      INPUT_OUTPUT::WRITE_STRING_-
TWO_VALUE_SP_SP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_VS(),      INPUT_-
OUTPUT::WRITE_STRING_TWO_VALUE_VS_C(),      INPUT_OUTPUT::WRITE_STRING_TWO_-
VALUE_VS_DP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_INTG(),      INPUT_-
OUTPUT::WRITE_STRING_TWO_VALUE_VS_L(),      INPUT_OUTPUT::WRITE_STRING_-
TWO_VALUE_VS_SP(),      INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_VS(),      INPUT_-
OUTPUT::WRITE_STRING_VALUE_C(),      INPUT_OUTPUT::WRITE_STRING_VALUE_DP(),
INPUT_OUTPUT::WRITE_STRING_VALUE_INTG(),      INPUT_OUTPUT::WRITE_STRING_-
VALUE_L(),      INPUT_OUTPUT::WRITE_STRING_VALUE_LINTG(),      INPUT_OUTPUT::WRITE_-
STRING_VALUE_SP(),      INPUT_OUTPUT::WRITE_STRING_VALUE_VS(),      and      INPUT_-
OUTPUT::WRITE_STRING_VS().

```

6.2.3.20 TYPE(ROUTINE_STACK_TYPE), save BASE_ROUTINES::ROUTINE_STACK

The routime invocation stack.

Definition at line 169 of file base_routines.f90.

6.2.3.21 LOGICAL, save BASE_ROUTINES::TIMING

.TRUE. if timing output is required in any routines.

Definition at line 161 of file base_routines.f90.

6.2.3.22 LOGICAL, save BASE_ROUTINES::TIMING_ALL_SUBROUTINES

.TRUE. if timing output is required in all routines

Definition at line 163 of file base_routines.f90.

6.2.3.23 LOGICAL,save BASE_ROUTINES::TIMING_FILE_OPEN

.TRUE. if the timing output file is open

Definition at line 165 of file base_routines.f90.

6.2.3.24 LOGICAL,save BASE_ROUTINES::TIMING_FROM_SUBROUTINE

.TRUE. if timing output is required from a particular routine

Definition at line 164 of file base_routines.f90.

6.2.3.25 TYPE(ROUTINE_LIST_TYPE),save BASE_ROUTINES::TIMING_ROUTINE_LIST

The list of routines for which timing output is required.

Definition at line 168 of file base_routines.f90.

6.2.3.26 LOGICAL,save BASE_ROUTINES::TIMING_SUMMARY

.TRUE. if timing output will be summary form via a TIMING_SUMMARY_OUTPUT call otherwise timing will be output for routines when the routine exits

See also:

[BASE_ROUTINES::TIMING_SUMMARY_OUTPUT](#)

Definition at line 162 of file base_routines.f90.

6.3 BINARY_FILE Namespace Reference

This module handles the reading and writing of binary files.

Classes

- struct [BINARY_FILE_INFO_TYPE](#)
- struct [BINARY_FILE_TYPE](#)
- struct [BINARY_TAG_TYPE](#)
- interface [interface](#)
- interface [READ_BINARY_FILE](#)
- interface [WRITE_BINARY_FILE](#)

Functions

- LOGICAL [INQUIRE_OPEN_BINARY_FILE](#) (FILEID)
- LOGICAL [INQUIRE_EOF_BINARY_FILE](#) (FILEID, ERR, ERROR)
- subroutine [CLOSE_BINARY_FILE](#) (FILEID, ERR, ERROR,*)
- subroutine [CLOSE_CMISS_BINARY_FILE](#) (FILEID, ERR, ERROR,*)
- subroutine [OPEN_BINARY_FILE](#) (FILEID, COMMAND, FILENAME, ERR, ERROR,*)
- subroutine [OPEN_CMISS_BINARY_FILE](#) (FILEID, FILE_TYPE, NUMBER_TA S,&VERSION, FILEVERSION, COMMAND, EXTENSION, FILENAME, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_INTG](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_INTG1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_SINTG](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_SINTG1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_LINTG](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_LINTG1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_SP](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_SP1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_DP](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_DP1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_CHARACTER](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_LOGICAL](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_LOGICAL1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_SPC](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_SPC1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_DPC](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_DPC1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_TAG_HEADER](#) (FILEID, TAG, ERR, ERROR,*)
- subroutine [RESET_BINARY_NUMBER_TAGS](#) (FILEID, NUMBER_TAGS, ERR, ERROR,*)
- subroutine [SET_BINARY_FILE](#) (FILEID, SET_CODE, ERR, ERROR,*)
- subroutine [SKIP_CM_BINARY_HEADER](#) (FILEID, SKIP, ERR, ERROR,*)
- subroutine [SKIP_BINARY_FILE](#) (FILEID, NUMBER_BYTES, ERR, ERROR,*)
- subroutine [SKIP_BINARY_TAGS](#) (FILEID, TAG, ERR, ERROR,*)
- subroutine [WRITE_BINARY_FILE_INTG](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [WRITE_BINARY_FILE_INTG1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [WRITE_BINARY_FILE_SINTG](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)

- subroutine `WRITE_BINARY_FILE_SINTG1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_LINTG` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_LINTG1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_SP` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_SP1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_DP` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_DP1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_CHARACTER` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_LOGICAL` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_LOGICAL1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_SPC` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_SPC1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_DPC` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_DPC1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_TAG_HEADER` (FILEID, TAG, ERR, ERROR,*)

Variables

- INTEGER(INTG), dimension, parameter `MAX_NUM_BINARY_FILES` = 99
- INTEGER(INTG), parameter `FILE_BEGINNING` = 0
- INTEGER(INTG), parameter `FILE_CURRENT` = 1
- INTEGER(INTG), parameter `FILE_END` = 2
- INTEGER(INTG), parameter `FILE SAME_ENDIAN` = 0
- INTEGER(INTG), parameter `FILE_CHANGE_ENDIAN` = 1
- INTEGER(INTG), parameter `BINARY_FILE_READABLE` = 1
- INTEGER(INTG), parameter `BINARY_FILE_WRITABLE` = 2
- INTEGER(INTG), parameter `CMISS_BINARY_IDENTITY` = 7
- INTEGER(INTG), parameter `CMISS_BINARY_MATRIX_FILE` = 1
- INTEGER(INTG), parameter `CMISS_BINARY_HISTORY_FILE` = 2
- INTEGER(INTG), parameter `CMISS_BINARY_SIGNAL_FILE` = 3
- INTEGER(INTG), parameter `CMISS_BINARY_IDENTITY_HEADER` = 1
- INTEGER(INTG), parameter `CMISS_BINARY_MACHINE_HEADER` = 2
- INTEGER(INTG), parameter `CMISS_BINARY_FILE_HEADER` = 3
- LOGICAL, save `BINARY_FILE_USED` = .FALSE.

6.3.1 Detailed Description

This module handles the reading and writing of binary files.

6.3.2 Function Documentation

6.3.2.1 subroutine `BINARY_FILE::CLOSE_BINARY_FILE` (TYPE(BINARY_- FILE_TYPE),intent(out) `FILEID`, INTEGER(INTG),intent(out) `ERR`, TYPE(VARYING_STRING),intent(out) `ERROR`, *)

Definition at line 474 of file `binary_file.f90`.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.3.2.2 subroutine BINARY_FILE::CLOSE_CMISS_BINARY_FILE (TYPE(BINARY_FILE_TYPE),intent(inout) FILEID, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 518 of file binary_file_f.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.3.2.3 LOGICAL BINARY_FILE::INQUIRE_EOF_BINARY_FILE (TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)

Definition at line 430 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.3.2.4 LOGICAL BINARY_FILE::INQUIRE_OPEN_BINARY_FILE (TYPE(BINARY_FILE_TYPE),intent(in) FILEID)

Definition at line 407 of file binary_file_f.f90.

6.3.2.5 subroutine BINARY_FILE::OPEN_BINARY_FILE (TYPE(BINARY_FILE_TYPE),intent(out) FILEID, CHARACTER(LEN=*),intent(in) COMMAND, CHARACTER(LEN=*),intent(in) FILENAME, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 547 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), F90C::F2CSTRING(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.3.2.6 subroutine BINARY_FILE::OPEN_CMISS_BINARY_FILE (TYPE(BINARY_FILE_TYPE),intent(out) FILEID, INTEGER(INTG),intent(in) FILE_TYPE, NUMBER_TAS, &,dimension(3),intent(in) VERSION, INTEGER(INTG),dimension(3),intent(out) FILEVERSION, CHARACTER(LEN=*),intent(in) COMMAND, CHARACTER(LEN=*),intent(in) EXTENSION, CHARACTER(LEN=*),intent(in) FILENAME, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 623 of file binary_file_f.f90.

References MACHINE_CONSTANTS::CHARACTER_SIZE, MACHINE_CONSTANTS::DOUBLE_COMPLEX_SIZE, MACHINE_CONSTANTS::DOUBLE_REAL_SIZE, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), BASE_ROUTINES::GENERAL_OUTPUT_TYPE, MACHINE_CONSTANTS::INTEGER_SIZE,

KINDS::INTG, MACHINE_CONSTANTS::LOGICAL_SIZE, MACHINE_CONSTANTS::LONG_INTEGER_SIZE, MACHINE_CONSTANTS::MACHINE_CHAR_FORMAT, MACHINE_CONSTANTS::MACHINE_DP_FORMAT, MACHINE_CONSTANTS::MACHINE_ENDIAN, MACHINE_CONSTANTS::MACHINE_INT_FORMAT, MACHINE_CONSTANTS::MACHINE_OS, MACHINE_CONSTANTS::MACHINE_SP_FORMAT, MACHINE_CONSTANTS::MACHINE_TYPE, BASE_ROUTINES::OP_STRING, MACHINE_CONSTANTS::SHORT_INTEGER_SIZE, MACHINE_CONSTANTS::SINGLE_COMPLEX_SIZE, and MACHINE_CONSTANTS::SINGLE_REAL_SIZE.

Here is the call graph for this function:

6.3.2.7 subroutine BINARY_FILE::READ_BINARY_FILE_CHARACTER
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, CHARACTER(LEN=*),intent(out) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 1341 of file binary_file_f.f90.

References F90C::C2FSTRING(), F90C::CSTRINGLENGTH(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.3.2.8 subroutine BINARY_FILE::READ_BINARY_FILE_DP (TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, REAL(DP),dimension(*),intent(out) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 1243 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.3.2.9 subroutine BINARY_FILE::READ_BINARY_FILE_DP1 (TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, REAL(DP),intent(out) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 1290 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.3.2.10 subroutine BINARY_FILE::READ_BINARY_FILE_DPC (TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, COMPLEX(DPC),dimension(*),intent(out) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 1591 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.3.2.11 subroutine BINARY_FILE::READ_BINARY_FILE_DPC1 (TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, COMPLEX(DPC),intent(out) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 1639 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.3.2.12 subroutine BINARY_FILE::READ_BINARY_FILE_INTG (TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, INTEGER(INTG),dimension(*),intent(out) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 851 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.3.2.13 subroutine BINARY_FILE::READ_BINARY_FILE_INTG1 (TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, INTEGER(INTG),intent(out) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 898 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.3.2.14 subroutine BINARY_FILE::READ_BINARY_FILE_LINTG (TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, INTEGER(LINTG),dimension(*),intent(out) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 1047 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

**6.3.2.15 subroutine `BINARY_FILE::READ_BINARY_FILE_LINTG1` (TYPE(BINARY_-
FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUM_DATA*,
INTEGER(LINTG),intent(out) *DATA*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 1094 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(),
BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

**6.3.2.16 subroutine `BINARY_FILE::READ_BINARY_FILE_LOGICAL`
(TYPE(BINARY_FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in)
NUM_DATA, LOGICAL,dimension(*),intent(out) *DATA*, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 1393 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(),
BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

**6.3.2.17 subroutine `BINARY_FILE::READ_BINARY_FILE_LOGICAL1`
(TYPE(BINARY_FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in)
NUM_DATA, LOGICAL,intent(out) *DATA*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 1440 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(),
BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

**6.3.2.18 subroutine `BINARY_FILE::READ_BINARY_FILE_SINTG` (TYPE(BINARY_-
FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUM_DATA*,
INTEGER(SINTG),dimension(*),intent(out) *DATA*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 949 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(),
BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

**6.3.2.19 subroutine `BINARY_FILE::READ_BINARY_FILE_SINTG1` (TYPE(BINARY_-
FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUM_DATA*,
INTEGER(SINTG),intent(out) *DATA*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 996 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.3.2.20 subroutine BINARY_FILE::READ_BINARY_FILE_SP (TYPE(BINARY_-FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, REAL(SP),dimension(*),intent(out) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 1145 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.3.2.21 subroutine BINARY_FILE::READ_BINARY_FILE_SP1 (TYPE(BINARY_-FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, REAL(SP),intent(out) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 1192 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.3.2.22 subroutine BINARY_FILE::READ_BINARY_FILE_SPC (TYPE(BINARY_-FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, COMPLEX(SPC),dimension(*),intent(out) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 1491 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.3.2.23 subroutine BINARY_FILE::READ_BINARY_FILE_SPC1 (TYPE(BINARY_-FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, COMPLEX(SPC),intent(out) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 1539 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

**6.3.2.24 subroutine `BINARY_FILE::READ_BINARY_TAG_HEADER` (TYPE(BINARY_-
FILE_TYPE),intent(in) *FILEID*, TYPE(BINARY_TAG_TYPE),intent(out) *TAG*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 1691 of file binary_file.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_-ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.3.2.25 subroutine `BINARY_FILE::RESET_BINARY_NUMBER_TAGS` (TYPE(BINARY_-
FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUMBER_TAGS*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 1749 of file binary_file.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_-ROUTINES::EXITS()`, `MACHINE_CONSTANTS::INTEGER_SIZE`, and `MACHINE_-CONSTANTS::SINGLE_REAL_SIZE`.

Here is the call graph for this function:

**6.3.2.26 subroutine `BINARY_FILE::SET_BINARY_FILE` (TYPE(BINARY_-
FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *SET_CODE*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 1803 of file binary_file.f90.

References `F90C::C2FSTRING()`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.3.2.27 subroutine `BINARY_FILE::SKIP_BINARY_FILE` (TYPE(BINARY_FILE_-
TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUMBER_BYTES*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 1918 of file binary_file.f90.

References `F90C::C2FSTRING()`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.3.2.28 subroutine `BINARY_FILE::SKIP_BINARY_TAGS` (TYPE(BINARY_FILE_-
TYPE),intent(in) *FILEID*, TYPE(BINARY_TAG_TYPE),intent(in) *TAG*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 1960 of file binary_file.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_-ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.3.2.29 subroutine BINARY_FILE::SKIP_CM_BINARY_HEADER (TYPE(BINARY_-
FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) SKIP,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)**

Definition at line 1846 of file binary_file_f.f90.

References MACHINE_CONSTANTS::CHARACTER_SIZE, BASE_ROUTINES::ENTERS(), BASE_-ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MACHINE_CONSTANTS::INTEGER_SIZE, and MACHINE_CONSTANTS::SINGLE_REAL_SIZE.

Here is the call graph for this function:

**6.3.2.30 subroutine BINARY_FILE::WRITE_BINARY_FILE_CHARACTER
(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)
NUM_DATA, CHARACTER(LEN=*),intent(in) DATA, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)**

Definition at line 2466 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and F90C::F2cstring().

Here is the call graph for this function:

**6.3.2.31 subroutine BINARY_FILE::WRITE_BINARY_FILE_DP (TYPE(BINARY_-
FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA,
REAL(DP),dimension(*),intent(in) DATA, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)**

Definition at line 2376 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.3.2.32 subroutine BINARY_FILE::WRITE_BINARY_FILE_DP1 (TYPE(BINARY_-
FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA,
REAL(DP),intent(in) DATA, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)**

Definition at line 2420 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.3.2.33 subroutine BINARY_FILE::WRITE_BINARY_FILE_DPC (TYPE(BINARY_-
FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA,
COMPLEX(DPC),dimension(*),intent(in) DATA, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)**

Definition at line 2689 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.3.2.34 subroutine `BINARY_FILE::WRITE_BINARY_FILE_DPC1` (TYPE(BINARY_-
FILE_TYPE),intent(in) `FILEID`, INTEGER(INTG),intent(in) `NUM_DATA`,
COMPLEX(DPC),intent(in) `DATA`, INTEGER(INTG),intent(out) `ERR`,
TYPE(VARYING_STRING),intent(out) `ERROR`, *)**

Definition at line 2733 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.3.2.35 subroutine `BINARY_FILE::WRITE_BINARY_FILE_INTG` (TYPE(BINARY_-
FILE_TYPE),intent(in) `FILEID`, INTEGER(INTG),intent(in) `NUM_DATA`,
INTEGER(INTG),dimension(*),intent(in) `DATA`, INTEGER(INTG),intent(out) `ERR`,
TYPE(VARYING_STRING),intent(out) `ERROR`, *)**

Definition at line 2022 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.3.2.36 subroutine `BINARY_FILE::WRITE_BINARY_FILE_INTG1` (TYPE(BINARY_-
FILE_TYPE),intent(in) `FILEID`, INTEGER(INTG),intent(in) `NUM_DATA`,
INTEGER(INTG),intent(in) `DATA`, INTEGER(INTG),intent(out) `ERR`,
TYPE(VARYING_STRING),intent(out) `ERROR`, *)**

Definition at line 2063 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.3.2.37 subroutine `BINARY_FILE::WRITE_BINARY_FILE_LINTG` (TYPE(BINARY_-
FILE_TYPE),intent(in) `FILEID`, INTEGER(INTG),intent(in) `NUM_DATA`,
INTEGER(LINTG),dimension(*),intent(in) `DATA`, INTEGER(INTG),intent(out) `ERR`,
TYPE(VARYING_STRING),intent(out) `ERROR`, *)**

Definition at line 2197 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.3.2.38 subroutine BINARY_FILE::WRITE_BINARY_FILE_LINTG1 (TYPE(BINARY_-FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, INTEGER(LINTG),intent(in) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 2240 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.3.2.39 subroutine BINARY_FILE::WRITE_BINARY_FILE_LOGICAL (TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, LOGICAL,dimension(*),intent(in) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 2510 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.3.2.40 subroutine BINARY_FILE::WRITE_BINARY_FILE_LOGICAL1 (TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, LOGICAL,intent(in) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 2552 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.3.2.41 subroutine BINARY_FILE::WRITE_BINARY_FILE_SINTG (TYPE(BINARY_-FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, INTEGER(SINTG),dimension(*),intent(in) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 2108 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.3.2.42 subroutine BINARY_FILE::WRITE_BINARY_FILE_SINTG1 (TYPE(BINARY_-FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, INTEGER(SINTG),intent(in) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 2151 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.3.2.43 subroutine `BINARY_FILE::WRITE_BINARY_FILE_SP` (TYPE(BINARY_-
FILE_TYPE),intent(in) `FILEID`, INTEGER(INTG),intent(in) `NUM_DATA`,
REAL(SP),dimension(*),intent(in) `DATA`, INTEGER(INTG),intent(out) `ERR`,
TYPE(VARYING_STRING),intent(out) `ERROR`, *)**

Definition at line 2286 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.3.2.44 subroutine `BINARY_FILE::WRITE_BINARY_FILE_SP1` (TYPE(BINARY_-
FILE_TYPE),intent(in) `FILEID`, INTEGER(INTG),intent(in) `NUM_DATA`,
REAL(SP),intent(in) `DATA`, INTEGER(INTG),intent(out) `ERR`,
TYPE(VARYING_STRING),intent(out) `ERROR`, *)**

Definition at line 2330 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.3.2.45 subroutine `BINARY_FILE::WRITE_BINARY_FILE_SPC` (TYPE(BINARY_-
FILE_TYPE),intent(in) `FILEID`, INTEGER(INTG),intent(in) `NUM_DATA`,
COMPLEX(SPC),dimension(*),intent(in) `DATA`, INTEGER(INTG),intent(out) `ERR`,
TYPE(VARYING_STRING),intent(out) `ERROR`, *)**

Definition at line 2598 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.3.2.46 subroutine `BINARY_FILE::WRITE_BINARY_FILE_SPC1` (TYPE(BINARY_-
FILE_TYPE),intent(in) `FILEID`, INTEGER(INTG),intent(in) `NUM_DATA`,
COMPLEX(SPC),intent(in) `DATA`, INTEGER(INTG),intent(out) `ERR`,
TYPE(VARYING_STRING),intent(out) `ERROR`, *)**

Definition at line 2642 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.3.2.47 subroutine BINARY_FILE::WRITE_BINARY_TAG_HEADER (TYPE(BINARY_-
FILE_TYPE),intent(in) FILEID, TYPE(BINARY_TAG_TYPE),intent(out) TAG,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)**

Definition at line 2780 of file binary_file_f.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.3.3 Variable Documentation

6.3.3.1 INTEGER(INTG),parameter BINARY_FILE::BINARY_FILE_READABLE = 1

Definition at line 249 of file binary_file_f.f90.

6.3.3.2 LOGICAL,save BINARY_FILE::BINARY_FILE_USED = .FALSE.

Definition at line 301 of file binary_file_f.f90.

6.3.3.3 INTEGER(INTG),parameter BINARY_FILE::BINARY_FILE_WRITABLE = 2

Definition at line 250 of file binary_file_f.f90.

6.3.3.4 INTEGER(INTG),parameter BINARY_FILE::CMISS_BINARY_FILE_HEADER = 3

Definition at line 259 of file binary_file_f.f90.

6.3.3.5 INTEGER(INTG),parameter BINARY_FILE::CMISS_BINARY_HISTORY_FILE = 2

Definition at line 255 of file binary_file_f.f90.

6.3.3.6 INTEGER(INTG),parameter BINARY_FILE::CMISS_BINARY_IDENTITY = 7

Definition at line 253 of file binary_file_f.f90.

**6.3.3.7 INTEGER(INTG),parameter BINARY_FILE::CMISS_BINARY_IDENTITY_HEADER
= 1**

Definition at line 257 of file binary_file_f.f90.

**6.3.3.8 INTEGER(INTG),parameter BINARY_FILE::CMISS_BINARY_MACHINE_HEADER
= 2**

Definition at line 258 of file binary_file_f.f90.

6.3.3.9 INTEGER(INTG),parameter BINARY_FILE::CMISS_BINARY_MATRIX_FILE = 1

Definition at line 254 of file binary_file_f.f90.

6.3.3.10 INTEGER(INTG),parameter BINARY_FILE::CMISS_BINARY_SIGNAL_FILE = 3

Definition at line 256 of file binary_file_f.f90.

6.3.3.11 INTEGER(INTG),parameter BINARY_FILE::FILE_BEGINNING = 0

Definition at line 240 of file binary_file_f.f90.

6.3.3.12 INTEGER(INTG),parameter BINARY_FILE::FILE_CHANGE_ENDIAN = 1

Definition at line 246 of file binary_file_f.f90.

6.3.3.13 INTEGER(INTG),parameter BINARY_FILE::FILE_CURRENT = 1

Definition at line 241 of file binary_file_f.f90.

6.3.3.14 INTEGER(INTG),parameter BINARY_FILE::FILE_END = 2

Definition at line 242 of file binary_file_f.f90.

6.3.3.15 INTEGER(INTG),parameter BINARY_FILE::FILE_SAME_ENDIAN = 0

Definition at line 245 of file binary_file_f.f90.

6.3.3.16 INTEGER(INTG),dimension,parameter BINARY_FILE::MAX_NUM_BINARY_FILES = 99

Definition at line 237 of file binary_file_f.f90.

6.4 BLAS Namespace Reference

This module contains the [interface](#) descriptions to the [BLAS](#) routines.

Classes

- interface [interface](#)

6.4.1 Detailed Description

This module contains the [interface](#) descriptions to the [BLAS](#) routines.

6.5 CLASSICAL_FIELD_ROUTINES Namespace Reference

This module handles all classical field class routines.

Functions

- subroutine `CL` (EQUATIONS_SET, EQUATIONS_TYPE, EQUATIONS_SUBTYPE,&ERR, ERROR,*)

Gets the problem type and subtype for a classical field equation set class.
- subroutine `CL` (EQUATIONS_SET, EQUATIONS_TYPE, EQUATIONS_SUBTYPE,&ERR, ERROR,*)

Sets/changes the problem type and subtype for a classical field equation set class.
- subroutine `CLASSICAL_FIELDFINITEELEMENTCALCULATE` (EQUATIONS_SET, ELEMENT_NUMBER, ERR, ERROR,*)

Calculates the element stiffness matrices and rhs vector for the given element number for a classical field class finite element equation set.
- subroutine `CLASSICAL_FIELDFINITEELEMENTJACOBIAN_EVALUATE` (EQUATIONS_SET, ELEMENT_NUMBER, ERR, ERROR,*)

Evaluates the element Jacobian matrix for the given element number for a classical field class finite element equation set.
- subroutine `CLASSICAL_FIELDFINITEELEMENTRESIDUAL_EVALUATE` (EQUATIONS_SET, ELEMENT_NUMBER, ERR, ERROR,*)

Evaluates the element residual and rhs vectors for the given element number for a classical field class finite element equation set.
- subroutine `CLASSICAL_FIELD_EQUATIONS_SET_SETUP` (EQUATIONS_SET, SETUP_TYPE, ACTION_TYPE, ERR, ERROR,*)

Sets up the equations set for a classical field equations set class.
- subroutine `CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_GET` (PROBLEM, PROBLEM_EQUATION_TYPE, PROBLEM_SUBTYPE, ERR, ERROR,*)

Gets the problem type and subtype for a classical field problem class.
- subroutine `CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_SET` (PROBLEM, PROBLEM_EQUATION_TYPE, PROBLEM_SUBTYPE, ERR, ERROR,*)

Sets/changes the problem type and subtype for a classical field problem class.
- subroutine `CLASSICAL_FIELD_PROBLEM_SETUP` (PROBLEM, SETUP_TYPE, ACTION_TYPE, ERR, ERROR,*)

Sets up the problem for a classical field problem class.

6.5.1 Detailed Description

This module handles all classical field class routines.

6.5.2 Function Documentation

6.5.2.1 subroutine CLASSICAL_FIELD_ROUTINES::CL (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, INTEGER(INTG),intent(in) *EQUATIONS_TYPE*, INTEGER(INTG),intent(in) *EQUATIONS_SUBTYPE*, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *) [private]

Sets/changes the problem type and subtype for a classical field equation set class.

Parameters:

EQUATIONS_SET A pointer to the equations set
EQUATIONS_TYPE The equation type
EQUATIONS_SUBTYPE The equation subtype

Definition at line 117 of file classical_field_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_ADVECTION_DIFFUSION_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_BIHARMONIC_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_DIFFUSION_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_HELMHOLTZ_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LAPLACE_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_POISSON_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_REACTION_DIFFUSION_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_WAVE_EQUATION_TYPE, BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_SUBTYPE_SET().

Here is the call graph for this function:

6.5.2.2 subroutine CLASSICAL_FIELD_ROUTINES::CL (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, INTEGER(INTG),intent(out) *EQUATIONS_TYPE*, INTEGER(INTG),intent(out) *EQUATIONS_SUBTYPE*, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *)

Gets the problem type and subtype for a classical field equation set class.

Parameters:

EQUATIONS_SET A pointer to the equations set
EQUATIONS_TYPE The equation type
EQUATIONS_SUBTYPE The equation subtype

Definition at line 81 of file classical_field_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_CLASSICAL_FIELD_CLASS, BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.5.2.3 subroutine CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SET_SETUP (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, INTEGER(INTG),intent(in) *SETUP_TYPE*, INTEGER(INTG),intent(in) *ACTION_TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Sets up the equations set for a classical field equations set class.

Parameters:

EQUATIONS_SET A pointer to the equations set
SETUP_TYPE The setup type
ACTION_TYPE The action type
ERR The error code
ERROR The error string

Definition at line 323 of file classical_field_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_ADVECTION_DIFFUSION_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_BIHARMONIC_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_DIFFUSION_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_HELMHOLTZ_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LAPLACE_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_POISSON_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_REACTION_DIFFUSION_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_WAVE_EQUATION_TYPE, BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_SETUP().

Here is the call graph for this function:

6.5.2.4 subroutine CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELDFINITE_ELEMENT_CALCULATE (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, INTEGER(INTG),intent(in) *ELEMENT_NUMBER*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Calculates the element stiffness matrices and rhs vector for the given element number for a clasical field class finite element equation set.

Parameters:

EQUATIONS_SET A pointer to the equations set
ELEMENT_NUMBER The element number to calcualate
ERR The error code
ERROR The error string

Definition at line 170 of file classical_field_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_ADVECTION_DIFFUSION_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_BIHARMONIC_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_DIFFUSION_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_HELMHOLTZ_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LAPLACE_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_POISSON_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_REACTION_DIFFUSION_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_WAVE_EQUATION_TYPE,

SET LAPLACE_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
 POISSON_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SETREACTION_-
 DIFFUSION_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_WAVE_-
 EQUATION_TYPE, BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and LAPLACE_-
 EQUATIONS_ROUTINES::LAPLACE_EQUATIONFINITE_ELEMENT_CALCULATE().

Here is the call graph for this function:

**6.5.2.5 subroutine CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELDFINITE_-
 ELEMENT_JACOBIAN_EVALUATE (TYPE(EQUATIONS_SET_TYPE),pointer
 EQUATIONS_SET, INTEGER(INTG),intent(in) ELEMENT_NUMBER,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)**

Evaluates the element Jacobian matrix for the given element number for a clasical field class finite element equation set.

Parameters:

EQUATIONS_SET A pointer to the equations set

ELEMENT_NUMBER The element number to evaluate the Jacobian for

ERR The error code

ERROR The error string

Definition at line 221 of file classical_field_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
 ADVECTION_DIFFUSION_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_-
 SET_BIHAMMORIC_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_-
 SET_DIFFUSION_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_-
 SET_HELMHOLTZ_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_-
 SET_LAPLACE_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
 POISSON_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SETREACTION_-
 DIFFUSION_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_WAVE_-
 EQUATION_TYPE, BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.5.2.6 subroutine CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELDFINITE_-
 ELEMENT_RESIDUAL_EVALUATE (TYPE(EQUATIONS_SET_TYPE),pointer
 EQUATIONS_SET, INTEGER(INTG),intent(in) ELEMENT_NUMBER,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)**

Evaluates the element residual and rhs vectors for the given element number for a clasical field class finite element equation set.

Parameters:

EQUATIONS_SET A pointer to the equations set

ELEMENT_NUMBER The element number to evaluate the residual for

ERR The error code

ERROR The error string

Definition at line 272 of file classical_field_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_ADVECTION_DIFFUSION_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_BIHARMONIC_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_DIFFUSION_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_HELMHOLTZ_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LAPLACE_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_POISSON_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_REACTION_DIFFUSION_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_WAVE_EQUATION_TYPE`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.5.2.7 subroutine CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_-
PROBLEM_CLASS_TYPE_GET (TYPE(PROBLEM_TYPE),pointer
PROBLEM, INTEGER(INTG),intent(out) PROBLEM_EQUATION_TYPE,
INTEGER(INTG),intent(out) PROBLEM_SUBTYPE, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)
```

Gets the problem type and subtype for a classical field problem class.

Parameters:

PROBLEM A pointer to the problem
PROBLEM_EQUATION_TYPE The problem type
PROBLEM_SUBTYPE The probom subtype
ERR The error code
ERROR The error string

Definition at line 375 of file classical_field_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `PROBLEM_CONSTANTS::PROBLEM_CLASSICAL_FIELD_CLASS`.

Referenced by `PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_GET_PTR()`.

Here is the call graph for this function:

Here is the caller graph for this function:

```
6.5.2.8 subroutine CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_-
PROBLEM_CLASS_TYPE_SET (TYPE(PROBLEM_TYPE),pointer
PROBLEM, INTEGER(INTG),intent(in) PROBLEM_EQUATION_TYPE,
INTEGER(INTG),intent(in) PROBLEM_SUBTYPE, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)
```

Sets/changes the problem type and subtype for a classical field problem class.

Parameters:

PROBLEM A pointer to the problem
PROBLEM_EQUATION_TYPE The problem type
PROBLEM_SUBTYPE The probom subtype

ERR The error code

ERROR The error string

Definition at line 410 of file classical_field_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SUBTYPE_SET()`, `PROBLEM_CONSTANTS::PROBLEM_ADVECTION_DIFFUSION_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_BIHARMONIC_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_DIFFUSION_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_HELMHOLTZ_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_LAPLACE_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_POISSON_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_REACTION_DIFFUSION_EQUATION_TYPE`, and `PROBLEM_CONSTANTS::PROBLEM_WAVE_EQUATION_TYPE`.

Referenced by `PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_PTR()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.5.2.9 subroutine `CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_SETUP` (TYPE(`PROBLEM_TYPE`),pointer **PROBLEM**, **INTEGER(INTG),intent(in)** `SETUP_TYPE`, **INTEGER(INTG),intent(in)** `ACTION_TYPE`, **INTEGER(INTG),intent(out)** `ERR`, **TYPE(VARYING_STRING),intent(out)** `ERROR`, *)

Sets up the problem for a classical field problem class.

Parameters:

PROBLEM A pointer to the problem

SETUP_TYPE The setup type

ACTION_TYPE The action type

ERR The error code

ERROR The error string

Definition at line 462 of file classical_field_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SETUP()`, `PROBLEM_CONSTANTS::PROBLEM_ADVECTION_DIFFUSION_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_BIHARMONIC_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_DIFFUSION_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_HELMHOLTZ_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_LAPLACE_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_POISSON_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_REACTION_DIFFUSION_EQUATION_TYPE`, and `PROBLEM_CONSTANTS::PROBLEM_WAVE_EQUATION_TYPE`.

Referenced by `PROBLEM_ROUTINES::PROBLEM_SETUP()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.6 CMISS Namespace Reference

The top level cmiss module.

Functions

- subroutine [CMISS_FINALISE](#) (ERR, ERROR,*)

Finalises CMISS.
- subroutine [CMISS_INITIALISE](#) (ERR, ERROR,*)

Initialises CMISS.
- subroutine [CMISS_WRITE_ERROR](#) (ERR, ERROR)

Writes the error string to screen.

Variables

- INTEGER(INTG), parameter [CMISS_MAJOR_VERSION](#) = 0
- INTEGER(INTG), parameter [CMISS_MINOR_VERSION](#) = 2
- INTEGER(INTG), parameter [CMISS_REVISION_VERSION](#) = 0
- CHARACTER(LEN=MAXSTRLEN), parameter [CMISS_BUILD_VERSION](#)

6.6.1 Detailed Description

The top level cmiss module.

6.6.2 Function Documentation

6.6.2.1 subroutine CMISS::CMISS_FINALISE (INTEGER(INTG),intent(inout) *ERR*, TYPE(VARYING_STRING),intent(inout) *ERROR*, *)

Finalises CMISS.

Parameters:

ERR The error string

ERROR The error code

Definition at line 95 of file cmiss.f90.

References [BASE_ROUTINES::BASE_ROUTINES_FINALISE\(\)](#), [COMP_-ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_FINALISE\(\)](#), [COORDINATE_-ROUTINES::COORDINATE_SYSTEMS_FINALISE\(\)](#), [GENERATED_MESH_-ROUTINES::GENERATED_MESHES_FINALISE\(\)](#), [PROBLEM_ROUTINES::PROBLEMS_FINALISE\(\)](#), and [REGION_ROUTINES::REGIONS_FINALISE\(\)](#).

Here is the call graph for this function:

**6.6.2.2 subroutine CMISS::CMISS_INITIALISE (INTEGER(INTG),intent(inout) *ERR*,
TYPE(VARYING_STRING),intent(inout) *ERROR*, *)**

Initialises [CMISS](#).

Parameters:

ERR The error code

ERROR The error string

Definition at line 126 of file cmiss.f90.

References `BASE_ROUTINES::BASE_ROUTINES_INITIALISE()`, `COMP_-
ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT`, `COMP_ENVIRONMENT::COMPUTATIONAL_-
ENVIRONMENT_INITIALISE()`, `COORDINATE_ROUTINES::COORDINATE_SYSTEMS_-
INITIALISE()`, `GENERATED_MESH_ROUTINES::GENERATED_MESHES_INITIALISE()`,
`PROBLEM_ROUTINES::PROBLEMS_INITIALISE()`, and `REGION_ROUTINES::REGIONS_-
INITIALISE()`.

Here is the call graph for this function:

**6.6.2.3 subroutine CMISS::CMISS_WRITE_ERROR (INTEGER(INTG),intent(inout) *ERR*,
TYPE(VARYING_STRING),intent(inout) *ERROR*)**

Writes the error string to screen.

Parameters:

ERR The error code

ERROR The error string

Definition at line 172 of file cmiss.f90.

References `MACHINE_CONSTANTS::ERROR_SEPARATOR_CONSTANT`.

6.6.3 Variable Documentation

6.6.3.1 CHARACTER(LEN=MAXSTRLEN),parameter CMISS::CMISS_BUILD_VERSION

Definition at line 76 of file cmiss.f90.

6.6.3.2 INTEGER(INTG),parameter CMISS::CMISS_MAJOR_VERSION = 0

Definition at line 72 of file cmiss.f90.

6.6.3.3 INTEGER(INTG),parameter CMISS::CMISS_MINOR_VERSION = 2

Definition at line 73 of file cmiss.f90.

6.6.3.4 INTEGER(INTG),parameter CMISS::CMISS_REVISION_VERSION = 0

Definition at line 74 of file cmiss.f90.

6.7 CMISS_MPI Namespace Reference

This module contains [CMISS](#) MPI routines.

Functions

- subroutine [MPI_ERROR_CHECK](#) (ROUTINE, MPI_ERR_CODE, ERR, ERROR,*)

Checks to see if an MPI error has occurred during an MPI call and flags a [CMISS](#) error if it has.

6.7.1 Detailed Description

This module contains [CMISS](#) MPI routines.

6.7.2 Function Documentation

6.7.2.1 subroutine CMISS_MPI::MPI_ERROR_CHECK (CHARACTER(LEN=*) *ROUTINE*, INTEGER(INTG),intent(in) *MPI_ERR_CODE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Checks to see if an MPI error has occurred during an MPI call and flags a [CMISS](#) error if it has.

Parameters:

ROUTINE The name of the MPI routine that has just been called.

MPI_ERR_CODE The MPI error code returned from the MPI routine.

ERR The error code.

ERROR The error string

Definition at line 74 of file cmiss_mpi.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_TYPE_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_TYPE_INITIALISE(), MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE(), FIELD_IO_ROUTINES::FIELD_IO_CREATE_FIELDS(), and FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH().

Here is the call graph for this function:

Here is the caller graph for this function:

6.8 CMISS_PARMETIS Namespace Reference

This module is a [CMISS](#) buffer module to the ParMETIS library.

Classes

- interface [interface](#)

Functions

- subroutine [PARMETIS_PARTKWAY](#) (VERTEX_DISTANCE, XADJ, ADJNCY, VERTEX_WEIGHT, ADJ_WEIGHT, WEIGHT_FLA, NUM_FLAG, NCON,&NUMBER_PARTS, TP_WEIGHTS, UB_VEC, OPTIONS, NUMBER_EDGES_CUT, PARTITION, COMMUNICATOR, ERR, ERROR,*)

Buffer routine to the ParMetis ParMETIS_V3_PartKway routine.

- subroutine [PARMETIS_PARTMESHKWAY](#) (ELEMENT_DISTANCE, ELEMENT_PTR, ELEMENT_INDEX, ELEMENT_WEIGHT, WEIGHT_FLAG, NUM_FLAG, CON,&NUMBER_COMMON_NODES, NUMBER_PARTS, TP_WEIGHTS, UB_VEC, OPTIONS, NUMBER_EDGES_CUT, PARTITION, COMMUNICATOR, ERR, ERROR,*)

Buffer routine to the ParMetis ParMETIS_V3_PartMeshKway routine.

6.8.1 Detailed Description

This module is a [CMISS](#) buffer module to the ParMETIS library.

6.8.2 Function Documentation

- 6.8.2.1 subroutine CMISS_PARMETIS::PARMETIS_PARTKWAY**
- ```
(INTEGER(INTG),dimension(:),intent(in) VERTEX_DISTANCE, INTEGER(INTG),dimension(:),intent(in) XADJ, INTEGER(INTG),dimension(:),intent(in) ADJNCY, INTEGER(INTG),dimension(:),intent(in) VERTEX_WEIGHT, INTEGER(INTG),dimension(:),intent(in) ADJ_WEIGHT, WEIGHT_FLA, INTEGER(INTG),intent(in) NUM_FLAG, INTEGER(INTG),intent(in) NCON, &,intent(in) NUMBER_PARTS, REAL(SP),dimension(:),intent(in) TP_WEIGHTS, REAL(SP),dimension(:),intent(in) UB_VEC, INTEGER(INTG),dimension(:),intent(in) OPTIONS, INTEGER(INTG),intent(out) NUMBER_EDGES_CUT, INTEGER(INTG),dimension(:),intent(out) PARTITION, INTEGER(INTG),intent(in) COMMUNICATOR, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)
```

Buffer routine to the ParMetis ParMETIS\_V3\_PartKway routine.

Definition at line 120 of file cmiss\_parmetis.f90.

References [BASE\\_ROUTINES::ENTERS\(\)](#), [BASE\\_ROUTINES::ERRORS\(\)](#), and [BASE\\_ROUTINES::EXITS\(\)](#).

Here is the call graph for this function:

```
6.8.2.2 subroutine CMISS_PARMETIS::PARMETIS_PARTMESHKWAY
 (INTEGER(INTG),dimension(:),intent(in) ELEMENT_DISTANCE,
 INTEGER(INTG),dimension(:),intent(in) ELEMENT_PTR,
 INTEGER(INTG),dimension(:),intent(in) ELEMENT_INDEX,
 INTEGER(INTG),dimension(:),intent(in) ELEMENT_WEIGHT,
 INTEGER(INTG),intent(in) WEIGHT_FLAG, INTEGER(INTG),intent(in)
 NUM_FLAG, CON, &,intent(in) NUMBER_COMMON_NODES,
 INTEGER(INTG),intent(in) NUMBER_PARTS, REAL(SP),dimension(:),intent(in)
 TP_WEIGHTS, REAL(SP),dimension(:),intent(in) UB_VEC,
 INTEGER(INTG),dimension(:),intent(in) OPTIONS, INTEGER(INTG),intent(out)
 NUMBER_EDGES_CUT, INTEGER(INTG),dimension(:),intent(out) PARTITION,
 INTEGER(INTG),intent(in) COMMUNICATOR, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *)
```

Buffer routine to the ParMetis ParMETIS\_V3\_PartMeshKway routine.

Definition at line 163 of file cmiss\_parmetis.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Referenced by MESH\_ROUTINES::DECOMPOSITION\_ELEMENT\_DOMAIN\_CALCULATE().

Here is the call graph for this function:

Here is the caller graph for this function:

## 6.9 CMISS\_PETSC Namespace Reference

This module is a [CMISS](#) buffer module to the PETSc library.

### Classes

- interface [interface](#)
- interface [PETSC\\_SNESSETJACOBIAN](#)

### Functions

- subroutine [PETSC\\_ERRORHANDLING\\_SET\\_OFF](#) (ERR, ERROR,\*)
 

*Set PETSc error handling on.*
- subroutine [PETSC\\_ERRORHANDLING\\_SET\\_ON](#) (ERR, ERROR,\*)
 

*Set PETSc error handling on.*
- subroutine [PETSC\\_FINALIZE](#) (ERR, ERROR,\*)
 

*Buffer routine to the PETSc PetscFinalize routine.*
- subroutine [PETSC\\_INITIALIZE](#) (FILE, ERR, ERROR,\*)
 

*Buffer routine to the PETSc PetscInitialize routine.*
- subroutine [PETSC\\_ISFINALISE](#) (IS\_, ERR, ERROR,\*)
 • subroutine [PETSC\\_ISINITIALISE](#) (IS\_, ERR, ERROR,\*)
 • subroutine [PETSC\\_ISDESTROY](#) (IS\_, ERR, ERROR,\*)
 

*Buffer routine to the PETSc ISDestroy routine.*
- subroutine [PETSC\\_ISCOLORINGFINALISE](#) (ISCOLORING, ERR, ERROR,\*)
 • subroutine [PETSC\\_ISCOLORINGINITIALISE](#) (ISCOLORING, ERR, ERROR,\*)
 • subroutine [PETSC\\_ISCOLORINGDESTROY](#) (ISCOLORING, ERR, ERROR,\*)
 

*Buffer routine to the PETSc ISColoringDestroy routine.*
- subroutine [PETSC\\_ISLOCALTOGLOBALMAPPINGFINALISE](#) (ISLOCALTOGLOBALMAPPING, ERR, ERROR,\*)
 • subroutine [PETSC\\_ISLOCALTOGLOBALMAPPINGINITIALISE](#) (ISLOCALTOGLOBALMAPPING, ERR, ERROR,\*)
 • subroutine [PETSC\\_ISLOCALTOGLOBALMAPPINGAPPLY](#) (CTX, TYPE, NIN, IDXIN, NOUT, IDXOUT, ERR, ERROR,\*)
 

*Buffer routine to the PETSc ISLocalToGlobalMappingApply routine.*
- subroutine [PETSC\\_ISLOCALTOGLOBALMAPPINGAPPLYIS](#) (CTX, ISIN, ISOOUT, ERR, ERROR,\*)
 

*Buffer routine to the PETSc ISLocalToGlobalMappingApplyIS routine.*
- subroutine [PETSC\\_ISLOCALTOGLOBALMAPPINGCREATE](#) (COMMUNICATOR, N, GLOBALNUM, CTX, ERR, ERROR,\*)
 

*Buffer routine to the PETSc ISLocalToGlobalMappingCreate routine.*

- subroutine [PETSC\\_ISLOCALTOGLOBALMAPPINGDESTROY](#) (CTX, ERR, ERROR,\*)
 

*Buffer routine to the PETSc ISLocalToGlobalMappingDestroy routine.*
- subroutine [PETSC\\_KSPCREATE](#) (COMMUNICATOR, KSP\_, ERR, ERROR,\*)
 

*Buffer routine to the PETSc KSPCreate routine.*
- subroutine [PETSC\\_KSPDESTROY](#) (KSP\_, ERR, ERROR,\*)
 

*Buffer routine to the PETSc KSPDestroy routine.*
- subroutine [PETSC\\_KSPGETCONVERGEDREASON](#) (KSP\_, REASON, ERR, ERROR,\*)
 

*Buffer routine to the PETSc KSPGetConvergedReason routine.*
- subroutine [PETSC\\_KSPGETITERATIONNUMBER](#) (KSP\_, ITERATION\_NUMBER, ERR, ERROR,\*)
 

*Buffer routine to the PETSc KSPGetIterationNumber routine.*
- subroutine [PETSC\\_KSPGETPC](#) (KSP\_, PC\_, ERR, ERROR,\*)
 

*Buffer routine to the PETSc KSPGetPC routine.*
- subroutine [PETSC\\_KSPGETRESIDUALNORM](#) (KSP\_, RESIDUAL\_NORM, ERR, ERROR,\*)
 

*Buffer routine to the PETSc KSPGetResidualNorm routine.*
- subroutine [PETSC\\_KSPFINALISE](#) (KSP\_, ERR, ERROR,\*)
   
 • subroutine [PETSC\\_KSPINITIALISE](#) (KSP\_, ERR, ERROR,\*)
   
 • subroutine [PETSC\\_KSPSETFROMOPTIONS](#) (KSP\_, ERR, ERROR,\*)
 

*Buffer routine to the PETSc KSPSetFromOptions routine.*
- subroutine [PETSC\\_KSPSETOPERATORS](#) (KSP\_, AMAT, PMAT, FLAG, ERR, ERROR,\*)
 

*Buffer routine to the PETSc KSPSetOperators routine.*
- subroutine [PETSC\\_KSPSETTOLERANCES](#) (KSP\_, RTOL, ATOL, DTOL, MAXITS, ERR, ERROR,\*)
 

*Buffer routine to the PETSc KSPSetTolerances routine.*
- subroutine [PETSC\\_KSPSETTYPE](#) (KSP\_, METHOD, ERR, ERROR,\*)
 

*Buffer routine to the PETSc KSPSetType routine.*
- subroutine [PETSC\\_KSPSETUP](#) (KSP\_, ERR, ERROR,\*)
 

*Buffer routine to the PETSc KSPSetUp routine.*
- subroutine [PETSC\\_KSPSOLVE](#) (KSP\_, B, X, ERR, ERROR,\*)
 

*Buffer routine to the PETSc KSPSolve routine.*
- subroutine [PETSC\\_LOGPRINTSUMMARY](#) (COMMUNICATOR, FILE, ERR, ERROR,\*)
 

*Buffer routine to the PETSc PetscLogPrintSummary routine.*
- subroutine [PETSC\\_MATASSEMBLYBEGIN](#) (A, ASSEMBLY\_TYPE, ERR, ERROR,\*)
 

*Buffer routine to the PETSc MatAssemblyBegin routine.*
- subroutine [PETSC\\_MATASSEMBLYEND](#) (A, ASSEMBLY\_TYPE, ERR, ERROR,\*)

*Buffer routine to the PETSc MatAssemblyEnd routine.*

- subroutine [PETSC\\_MATCREATE](#) (COMMUNICATOR, A, ERR, ERROR,\*)

*Buffer routine to the PETSc MatCreate routine.*

- subroutine [PETSC\\_MATCREATEMPIAIJ](#) (COMMUNICATOR, LOCAL\_M, LOCAL\_N, GLOB\_L\_M, GLOBAL\_N, DIAG\_NUMBER\_NZ\_PERROW, DIAG\_NUMBER\_NZ\_EACHROW,&OFFDIAG\_NUMBER\_NZ\_PERROW, OFFDIAG\_NUMBER\_NZ\_EACHROW, A, ERR, ERROR,\*)

*Buffer routine to the PETSc MatCreateMPIAIJ routine.*

- subroutine [PETSC\\_MATCREATEMPIDENSE](#) (COMMUNICATOR, LOCAL\_M, LOCAL\_N, GLOBAL\_M, GLOBAL\_N, MATRIX\_DATA, A, ERR, ERROR,\*)

*Buffer routine to the PETSc MatCreateMPIDense routine.*

- subroutine [PETSC\\_MATCREATESEQAIJ](#) (COMMUNICATOR, M, N, NUMBER\_NZ\_PERROW, NUMBER\_NZ\_EACHROW, A, ERR, ERROR,\*)

*Buffer routine to the PETSc MatCreateSeqAIJ routine.*

- subroutine [PETSC\\_MATCREATESEQDENSE](#) (COMMUNICATOR, M, N, MATRIX\_DATA, A, ERR, ERROR,\*)

*Buffer routine to the PETSc MatCreateSeqDense routine.*

- subroutine [PETSC\\_MATDESTROY](#) (A, ERR, ERROR,\*)

*Buffer routine to the PETSc MatDestroy routine.*

- subroutine [PETSC\\_MATFDCOLORINGCREATE](#) (A, ISCOLORING, FDCOLORING, ERR, ERROR,\*)

*Buffer routine to the PETSc MatFDColoringCreate routine.*

- subroutine [PETSC\\_MATFDCOLORINGDESTROY](#) (MATFDCOLORING, ERR, ERROR,\*)

*Buffer routine to the PETSc MatFDColoringDestroy routine.*

- subroutine [PETSC\\_MATFDCOLORINGFINALISE](#) (MATFDCOLORING, ERR, ERROR,\*)

- subroutine [PETSC\\_MATFDCOLORINGINITIALISE](#) (MATFDCOLORING, ERR, ERROR,\*)

- subroutine [PETSC\\_MATFDCOLORINGSETFROMOPTIONS](#) (MATFDCOLORING, ERR, ERROR,\*)

*Buffer routine to the PETSc MatFDColoringSetFromOptions routine.*

- subroutine [PETSC\\_MATGETARRAY](#) (A, ARRAY, ERR, ERROR,\*)

*Buffer routine to the PETSc MatGetArray routine.*

- subroutine [PETSC\\_MATGETCOLORING](#) (A, COLORING\_TYPE, ISCOLORING, ERR, ERROR,\*)

*Buffer routine to the PETSc MatGetColoring routine.*

- subroutine [PETSC\\_MATGETOWNERSHIPRANGE](#) (A, FIRST\_ROW, LAST\_ROW, ERR, ERROR,\*)

*Buffer routine to the PETSc MatGetOwnershipRange routine.*

- subroutine [PETSC\\_MATGETVALUES](#) (A, M, M\_INDICES, N, N\_INDICES, VALUES, ERR, ERROR,\*)

*Buffer routine to the PETSc MatGetValues routine.*

- subroutine [PETSC\\_MATFINALISE](#) (MAT\_, ERR, ERROR,\*)
- subroutine [PETSC\\_MATINITIALISE](#) (MAT\_, ERR, ERROR,\*)
- subroutine [PETSC\\_MATRESTOREARRAY](#) (A, ERR, ERROR,\*)

*Buffer routine to the PETSc MatRestoreArray routine.*

- subroutine [PETSC\\_MATSETLOCALTOGLOBALMAPPING](#) (A, CTX, ERR, ERROR,\*)

*Buffer routine to the PETSc MatSetLocalToGlobalMapping routine.*

- subroutine [PETSC\\_MATSETOPTION](#) (A, OPTION, ERR, ERROR,\*)

*Buffer routine to the PETSc MatSetOption routine.*

- subroutine [PETSC\\_MATSETSIZES](#) (A, LOCAL\_M, LOCAL\_N, GLOBAL\_M, GLOBAL\_N, ERR, ERROR,\*)

*Buffer routine to the PETSc MatSetSizes routine.*

- subroutine [PETSC\\_MATSETVALUE](#) (A, ROW, COL, VALUE, INSERT\_MODE, ERR, ERROR,\*)

*Buffer routine to the PETSc MatSetValue routine.*

- subroutine [PETSC\\_MATSETVALUES](#) (A, M, M\_INDICES, N, N\_INDICES, VALUES, INSERT\_MODE, ERR, ERROR,\*)

*Buffer routine to the PETSc MatSetValues routine.*

- subroutine [PETSC\\_MATSETVALUELOCAL](#) (A, ROW, COL, VALUE, INSERT\_MODE, ERR, ERROR,\*)

*Buffer routine to the PETSc MatSetValueLocal routine.*

- subroutine [PETSC\\_MATSETVALUESLOCAL](#) (A, M, M\_INDICES, N, N\_INDICES, VALUES, INSERT\_MODE, ERR, ERROR,\*)

*Buffer routine to the PETSc MatSetValuesLocal routine.*

- subroutine [PETSC\\_MATVIEW](#) (A, V, ERR, ERROR,\*)

*Buffer routine to the PETSc MatView routine.*

- subroutine [PETSC\\_MATZEROENTRIES](#) (A, ERR, ERROR,\*)

*Buffer routine to the PETSc MatZeroEntries routine.*

- subroutine [PETSC\\_PCFINALISE](#) (PC\_, ERR, ERROR,\*)

- subroutine [PETSC\\_PCINITIALISE](#) (PC\_, ERR, ERROR,\*)

- subroutine [PETSC\\_PCSETTYPE](#) (PC\_, METHOD, ERR, ERROR,\*)

*Buffer routine to the PETSc PCSetType routine.*

- subroutine [PETSC\\_SNESFINALISE](#) (SNES\_, ERR, ERROR,\*)

- subroutine [PETSC\\_SNESINITIALISE](#) (SNES\_, ERR, ERROR,\*)

- subroutine [PETSC\\_SNESCREATE](#) (COMMUNICATOR, SNES\_, ERR, ERROR,\*)

*Buffer routine to the PETSc SNESCreate routine.*

- subroutine [PETSC\\_SNESDESTROY](#) (SNES\_, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESDestroy routine.*
- subroutine [PETSC\\_SNESGETCONVERGEDREASON](#) (SNES\_, REASON, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESGetConvergedReason routine.*
- subroutine [PETSC\\_SNESGETFUNCTIONNORM](#) (SNES\_, FUNCTION\_NORM, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESGetFunctionNorm routine.*
- subroutine [PETSC\\_SNESGETITERATIONNUMBER](#) (SNES\_, ITERATION\_NUMBER, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESGetIterationNumber routine.*
- subroutine [PETSC\\_SNESLINESEARCHSET](#) (SNES\_, LINESEARCH\_TYPE, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESLineSearchSet routine.*
- subroutine [PETSC\\_SNESLINESEARCHSETPARAMS](#) (SNES\_, ALPHA, MAXSTEP, STEPTOL, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESLineSearchSetParams routine.*
- subroutine [PETSC\\_SNESSETFROMOPTIONS](#) (SNES\_, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESSetFromOptions routine.*
- subroutine [PETSC\\_SNESSETFUNCTION](#) (SNES\_, F, FFUNCTION, CTX, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESSetFunction routine.*
- subroutine [PETSC\\_SNESSETJACOBIAN\\_MATFDCOLORING](#) (SNES\_, A, B, JFUNCTION, CTX, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESSetJacobian routine for MatFDColoring contexts.*
- subroutine [PETSC\\_SNESSETJACOBIAN\\_SOLVER](#) (SNES\_, A, B, JFUNCTION, CTX, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESSetJacobian routine for solver contexts.*
- subroutine [PETSC\\_SNESSETTOLERANCES](#) (SNES\_, ABSTOL, RTOL, STOL, MAXIT, MAXF, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESSetTolerances routine.*
- subroutine [PETSC\\_SNESSETTRUSTREGIONTOLERANCE](#) (SNES\_, TRTOL, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESSetTrustRegionTolerance routine.*
- subroutine [PETSC\\_SNESSETTYPE](#) (SNES\_, METHOD, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESSetType routine.*
- subroutine [PETSC\\_SNESSOLVE](#) (SNES\_, B, X, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESSolve routine.*

- subroutine [PETSC\\_VECFINALISE](#) (VEC\_, ERR, ERROR,\*)
- subroutine [PETSC\\_VECINITIALISE](#) (VEC\_, ERR, ERROR,\*)
- subroutine [PETSC\\_VECASSEMBLYBEGIN](#) (X, ERR, ERROR,\*)

*Buffer routine to the PETSc VecAssemblyBegin routine.*

- subroutine [PETSC\\_VECASSEMBLYEND](#) (X, ERR, ERROR,\*)

*Buffer routine to the PETSc VecAssemblyEnd routine.*

- subroutine [PETSC\\_VECCREATE](#) (COMMUNICATOR, X, ERR, ERROR,\*)

*Buffer routine to the PETSc VecCreate routine.*

- subroutine [PETSC\\_VECCREATEGHOST](#) (COMMUNICATOR, LOCAL\_SIZE, GLOBAL\_SIZE, NUMBER\_GHOST, GHOSTS, X, ERR, ERROR,\*)

*Buffer routine to the PETSc VecCreateGhost routine.*

- subroutine [PETSC\\_VECCREATEGHOSTWITHARRAY](#) (COMMUNICATOR, LOCAL\_SIZE, GLOBAL\_SIZE, NUMBER\_GHOST, GHOSTS, ARRAY, X, ERR, ERROR,\*)

*Buffer routine to the PETSc VecCreateGhostWithArray routine.*

- subroutine [PETSC\\_VECCREATEMPI](#) (COMMUNICATOR, LOCAL\_SIZE, GLOBAL\_SIZE, X, ERR, ERROR,\*)

*Buffer routine to the PETSc VecCreateMPI routine.*

- subroutine [PETSC\\_VECCREATEMPIWITHARRAY](#) (COMMUNICATOR, LOCAL\_SIZE, GLOBAL\_SIZE, ARRAY, X, ERR, ERROR,\*)

*Buffer routine to the PETSc VecCreateMPIWithArray routine.*

- subroutine [PETSC\\_VECCREATESEQ](#) (COMMUNICATOR, SIZE, X, ERR, ERROR,\*)

*Buffer routine to the PETSc VecCreateSeq routine.*

- subroutine [PETSC\\_VECCREATESEQWITHARRAY](#) (COMMUNICATOR, SIZE, ARRAY, X, ERR, ERROR,\*)

*Buffer routine to the PETSc VecCreateSeqWithArray routine.*

- subroutine [PETSC\\_VECDESTROY](#) (X, ERR, ERROR,\*)

*Buffer routine to the PETSc VecDestroy routine.*

- subroutine [PETSC\\_VECDUPLICATE](#) (OLD, NEW, ERR, ERROR,\*)

*Buffer routine to the PETSc VecDuplicate routine.*

- subroutine [PETSC\\_VECGETARRAY](#) (X, ARRAY, ERR, ERROR,\*)

*Buffer routine to the PETSc VecGetArray routine.*

- subroutine [PETSC\\_VECGETARRAYF90](#) (X, ARRAY, ERR, ERROR,\*)

*Buffer routine to the PETSc VecGetArrayF90 routine.*

- subroutine [PETSC\\_VECGETLOCALSIZE](#) (X, SIZE, ERR, ERROR,\*)

*Buffer routine to the PETSc VecGetLocalSize routine.*

- subroutine [PETSC\\_VECGETOWNERSHIPRANGE](#) (X, LOW, HIGH, ERR, ERROR,\*)

*Buffer routine to the PETSc VecGetOwnershipRange routine.*

- subroutine [PETSC\\_VECGETSIZE](#) (X, SIZE, ERR, ERROR,\*)

*Buffer routine to the PETSc VecGetSize routine.*

- subroutine [PETSC\\_VECGETVALUES](#) (X, N, INDICES, VALUES, ERR, ERROR,\*)

*Buffer routine to the PETSc VecGetValues routine.*

- subroutine [PETSC\\_VECGHOSTGETLOCALFORM](#) (G, L, ERR, ERROR,\*)

*Buffer routine to the PETSc VecGhostGetLocalForm routine.*

- subroutine [PETSC\\_VECGHOSTRESTORELOCALFORM](#) (G, L, ERR, ERROR,\*)

*Buffer routine to the PETSc VecGhostRestoreLocalForm routine.*

- subroutine [PETSC\\_VECGHOSTUPDATEBEGIN](#) (X, INSERT\_MODE, SCATTER\_MODE, ERR, ERROR,\*)

*Buffer routine to the PETSc VecGhostUpdateBegin routine.*

- subroutine [PETSC\\_VECGHOSTUPDATEEND](#) (X, INSERT\_MODE, SCATTER\_MODE, ERR, ERROR,\*)

*Buffer routine to the PETSc VecGhostUpdateEnd routine.*

- subroutine [PETSC\\_VCRESTOREARRAY](#) (X, ERR, ERROR,\*)

*Buffer routine to the PETSc VecRestoreArray routine.*

- subroutine [PETSC\\_VCRESTOREARRAYF90](#) (X, ARRAY, ERR, ERROR,\*)

*Buffer routine to the PETSc VecRestoreArrayF90 routine.*

- subroutine [PETSC\\_VECSET](#) (X, VALUE, ERR, ERROR,\*)

*Buffer routine to the PETSc VecSet routine.*

- subroutine [PETSC\\_VECSETFROMOPTIONS](#) (X, ERR, ERROR,\*)

*Buffer routine to the PETSc VecSetFromOptions routine.*

- subroutine [PETSC\\_VECSETLOCALTOGLOBALMAPPING](#) (X, CTX, ERR, ERROR,\*)

*Buffer routine to the PETSc VecSetLocalToGlobalMapping routine.*

- subroutine [PETSC\\_VECSETVALUES](#) (X, N, INDICES, VALUES, INSERT\_MODE, ERR, ERROR,\*)

*Buffer routine to the PETSc VecSetValues routine.*

- subroutine [PETSC\\_VECSETVALUESLOCAL](#) (X, N, INDICES, VALUES, INSERT\_MODE, ERR, ERROR,\*)

*Buffer routine to the PETSc VecSetValuesLocal routine.*

- subroutine [PETSC\\_VECSETSIZES](#) (X, LOCAL\_SIZE, GLOBAL\_SIZE, ERR, ERROR,\*)

*Buffer routine to the PETSc VecSetSizes routine.*

- subroutine [PETSC\\_VECVIEW](#) (X, V, ERR, ERROR,\*)

*Buffer routine to the PETSc VecView routine.*

## Variables

- INTEGER(INTG), parameter `PETSC_SNES_LINESEARCH_NONORMS` = 1
- INTEGER(INTG), parameter `PETSC_SNES_LINESEARCH_NO` = 2
- INTEGER(INTG), parameter `PETSC_SNES_LINESEARCH_QUADRATIC` = 3
- INTEGER(INTG), parameter `PETSC_SNES_LINESEARCH_CUBIC` = 4
- LOGICAL, save `PETSC_HANDLE_ERROR`

### 6.9.1 Detailed Description

This module is a `CMISS` buffer module to the PETSc library.

### 6.9.2 Function Documentation

#### 6.9.2.1 subroutine CMISS\_PETSC::PETSC\_ERRORHANDLING\_SET\_OFF (INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Set PETSc error handling on.

##### Parameters:

*ERR* The error code

*ERROR* The error string

Definition at line 922 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

#### 6.9.2.2 subroutine CMISS\_PETSC::PETSC\_ERRORHANDLING\_SET\_ON (INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Set PETSc error handling on.

##### Parameters:

*ERR* The error code

*ERROR* The error string

Definition at line 945 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

#### 6.9.2.3 subroutine CMISS\_PETSC::PETSC\_FINALIZE (INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Buffer routine to the PETSc `PetscFinalize` routine.

**Parameters:*****ERR*** The error code***ERROR*** The error string

Definition at line 968 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.9.2.4 subroutine CMISS\_PETSC::PETSC\_INITIALIZE (CHARACTER(LEN=\*),intent(in) *FILE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Buffer routine to the PETSc PetscInitialize routine.

**Parameters:*****FILE*** Filename for PETSc options file***ERR*** The error code***ERROR*** The error string

Definition at line 997 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.9.2.5 subroutine CMISS\_PETSC::PETSC\_ISCOLORINGDESTROY (TYPE(PETSC\_ISCOLORING\_TYPE),intent(in) *ISCOLORING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Buffer routine to the PETSc ISColoringDestroy routine.

**Parameters:*****ISCOLORING*** The index set coloring***ERR*** The error code***ERROR*** The error string

Definition at line 1159 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.9.2.6 subroutine CMISS\_PETSC::PETSC\_ISCOLORINGFINALISE (TYPE(PETSC\_ISCOLORING\_TYPE),intent(inout) *ISCOLORING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:**

***ISCOLORING*** The ISColoring to finalise

***ERR*** The error code

***ERROR*** The error string

Definition at line 1109 of file cmiss\_petsc.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    and    **BASE\_ROUTINES::EXITS()**.

Referenced by **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_FINALISE()**.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.9.2.7 subroutine CMISS\_PETSC::PETSC\_ISCOLORINGINITIALISE (TYPE(PETSC\_ISCOLORING\_TYPE),intent(inout) *ISCOLORING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:**

***ISCOLORING*** The ISColoring to initialise

***ERR*** The error code

***ERROR*** The error string

Definition at line 1135 of file cmiss\_petsc.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    and    **BASE\_ROUTINES::EXITS()**.

Referenced by **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_INITIALISE()**.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.9.2.8 subroutine CMISS\_PETSC::PETSC\_ISDESTROY (TYPE(PETSC\_IS\_TYPE),intent(in) *IS\_*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Buffer routine to the PETSc ISDestroy routine.

**Parameters:**

***IS\_*** The index set

***ERR*** The error code

**ERROR** The error string

Definition at line 1078 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.9 subroutine CMISS\_PETSC::PETSC\_ISFINALISE (TYPE(PETSC\_IS\_TYPE),intent(inout) IS\_, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

**Parameters:**

*IS\_* The IS to finalise

*ERR* The error code

*ERROR* The error string

Definition at line 1028 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.10 subroutine CMISS\_PETSC::PETSC\_ISINITIALISE (TYPE(PETSC\_IS\_TYPE),intent(inout) IS\_, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

**Parameters:**

*IS\_* The IS to initialise

*ERR* The error code

*ERROR* The error string

Definition at line 1054 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.11 subroutine CMISS\_PETSC::PETSC\_ISLOCALTOGLOBALMAPPINGAPPLY (TYPE(PETSC\_ISLOCALTOGLOBALMAPPING\_TYPE),intent(in) CTX, INTEGER(INTG),intent(in) TYPE, INTEGER(INTG),intent(in) NIN, INTEGER(INTG),dimension(\*),intent(in) IDXIN, INTEGER(INTG),intent(out) NOUT, INTEGER(INTG),dimension(\*),intent(out) IDXOUT, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Buffer routine to the PETSc ISLocalToGlobalMappingApply routine.

**Parameters:**

*CTX* The local to global mapping context

**TYPE** The type of local to global mapping

**NIN** The number of local indices

**ERR** The error code

**ERROR** The error string

Definition at line 1238 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.12 subroutine CMISS\_PETSC::PETSC\_ISLOCALTOGLOBALMAPPINGAPPLYIS**  
`(TYPE(PETSC_ISLOCALTOGLOBALMAPPING_TYPE),intent(in) CTX,`  
`TYPE(PETSC_IS_TYPE),intent(in) ISIN, TYPE(PETSC_IS_TYPE),intent(out) ISOUT,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Buffer routine to the PETSc ISLocalToGlobalMappingApplyIS routine.

#### Parameters:

**CTX** The local to global mapping context

**ERR** The error code

**ERROR** The error string

Definition at line 1273 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.13 subroutine CMISS\_PETSC::PETSC\_ISLOCALTOGLOBALMAPPINGCREATE**  
`(COMMUNICATOR, INTEGER(INTG),intent(in) N,`  
`INTEGER(INTG),dimension(*),intent(in) GLOBALNUM,`  
`TYPE(PETSC_ISLOCALTOGLOBALMAPPING_TYPE),intent(inout) CTX,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Buffer routine to the PETSc ISLocalToGlobalMappingCreate routine.

#### Parameters:

**N** The number of local indices

**GLOBALNUM** The global number for each local index

**CTX** The local to global mapping context

**ERR** The error code

**ERROR** The error string

Definition at line 1305 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

---

**6.9.2.14 subroutine CMISS\_PETSC::PETSC\_ISLOCALTOGLOBALMAPPINGDESTROY**  
 (TYPE(PETSC\_ISLOCALTOGLOBALMAPPING\_TYPE),intent(inout) *CTX*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Buffer routine to the PETSc ISLocalToGlobalMappingDestroy routine.

**Parameters:**

*CTX* The local to global mapping context  
*ERR* The error code  
*ERROR* The error string

Definition at line 1338 of file cmiss\_petsc.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.15 subroutine CMISS\_PETSC::PETSC\_ISLOCALTOGLOBALMAPPINGFINALISE**  
 (TYPE(PETSC\_ISLOCALTOGLOBALMAPPING\_TYPE),intent(inout)  
*ISLOCALTOGLOBALMAPPING*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:**

*ISLOCALTOGLOBALMAPPING* The ISLocalToGlobalMapping to finalise  
*ERR* The error code  
*ERROR* The error string

Definition at line 1188 of file cmiss\_petsc.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.16 subroutine CMISS\_PETSC::PETSC\_ISLOCALTOGLOBALMAPPINGINITIALISE**  
 (TYPE(PETSC\_ISLOCALTOGLOBALMAPPING\_TYPE),intent(inout)  
*ISLOCALTOGLOBALMAPPING*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:**

*ISLOCALTOGLOBALMAPPING* The ISLocalToGlobalMapping to initialise  
*ERR* The error code  
*ERROR* The error string

Definition at line 1214 of file cmiss\_petsc.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

---

**6.9.2.17 subroutine CMISS\_PETSC::PETSC\_KSPCREATE (COMMUNICATOR,  
TYPE(PETSC\_KSP\_TYPE),intent(inout) KSP\_, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Buffer routine to the PETSc KSPCreate routine.

**Parameters:**

**KSP\_** On exit, the Ksp information

**ERR** The error code

**ERROR** The error string

Definition at line 1368 of file cmiss\_petsc.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Referenced by SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_CREATE\_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.9.2.18 subroutine CMISS\_PETSC::PETSC\_KSPDESTROY (TYPE(PETSC\_-  
KSP\_TYPE),intent(inout) KSP\_, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Buffer routine to the PETSc KSPDestroy routine.

**Parameters:**

**KSP\_** The Ksp to destroy

**ERR** The error code

**ERROR** The error string

Definition at line 1399 of file cmiss\_petsc.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.19 subroutine CMISS\_PETSC::PETSC\_KSPFINALISE (TYPE(PETSC\_-  
KSP\_TYPE),intent(inout) KSP\_, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

**Parameters:**

**KSP\_** The Ksp to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 1553 of file cmiss\_petsc.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Referenced by SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.9.2.20 subroutine CMISS\_PETSC::PETSC\_KSPGETCONVERGEDREASON**  
`(TYPE(PETSC_KSP_TYPE),intent(inout) KSP_, INTEGER(INTG),intent(out) REASON, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Buffer routine to the PETSc KSPGetConvergedReason routine.

**Parameters:**

*KSP\_* The KSP information

*REASON* On exit, the converged reason

*ERR* The error code

*ERROR* The error string

Definition at line 1429 of file cmiss\_petsc.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.9.2.21 subroutine CMISS\_PETSC::PETSC\_KSPGETITERATIONNUMBER** `(TYPE(PETSC_KSP_TYPE),intent(inout) KSP_, INTEGER(INTG),intent(out) ITERATION_NUMBER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`  
`[private]`

Buffer routine to the PETSc KSPGetIterationNumber routine.

**Parameters:**

*KSP\_* The KSP information

*ITERATION\_NUMBER* On exit, the number of iterations

*ERR* The error code

*ERROR* The error string

Definition at line 1460 of file cmiss\_petsc.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.9.2.22 subroutine CMISS\_PETSC::PETSC\_KSPGETPC (TYPE(PETSC\_KSP\_-  
TYPE),intent(inout) *KSP\_*, TYPE(PETSC\_PC\_TYPE),intent(inout) *PC\_*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc KSPGetPC routine.

**Parameters:**

***KSP\_*** The Ksp to get the PC for  
***PC\_*** On exit, the PC associated with the Ksp  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 1491 of file cmiss\_petsc.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Referenced by SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_CREATE\_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.9.2.23 subroutine CMISS\_PETSC::PETSC\_KSPGETRESIDUALNORM (TYPE(PETSC\_KSP\_TYPE),intent(inout) *KSP\_*, REAL(DP),intent(out) *RESIDUAL\_NORM*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc KSPGetResidualNorm routine.

**Parameters:**

***KSP\_*** The Ksp to get the PC for  
***RESIDUAL\_NORM*** On exit, the residual norm for the KSP  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 1522 of file cmiss\_petsc.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Referenced by SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.9.2.24 subroutine CMISS\_PETSC::PETSC\_KSPINITIALISE (TYPE(PETSC\_KSP\_TYPE),intent(inout) *KSP\_*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

**Parameters:**

***KSP\_*** The Ksp to initialise  
***ERR*** The error code

**ERROR** The error string

Definition at line 1579 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.9.2.25 subroutine CMISS\_PETSC::PETSC\_KSPSETFROMOPTIONS**  
`(TYPE(PETSC_KSP_TYPE),intent(inout) KSP_, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Buffer routine to the PETSc KSPSetFromOptions routine.

**Parameters:**

**KSP\_** The Ksp to set the options for

**ERR** The error code

**ERROR** The error string

Definition at line 1603 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.9.2.26 subroutine CMISS\_PETSC::PETSC\_KSPSETOPERATORS** `(TYPE(PETSC_KSP_TYPE),intent(inout) KSP_, TYPE(PETSC_MAT_TYPE),intent(inout) AMAT,`  
`TYPE(PETSC_MAT_TYPE),intent(inout) PMAT, INTEGER(INTG),intent(in) FLAG,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Buffer routine to the PETSc KSPSetOperators routine.

**Parameters:**

**KSP\_** The Ksp to set the operators for

**AMAT** The matrix associated with the linear system

**PMAT** The matrix to be used in constructing the preconditioner

**FLAG** Preconditioner matrix structure flag

**ERR** The error code

**ERROR** The error string

Definition at line 1633 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### **6.9.2.27 subroutine CMISS\_PETSC::PETSC\_KSPSETTOLERANCES (TYPE(PETSC\_KSP\_TYPE),intent(inout) KSP\_, REAL(DP),intent(in) RTOL, REAL(DP),intent(in) ATOL, REAL(DP),intent(in) DTOL, INTEGER(INTG),intent(in) MAXITS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Buffer routine to the PETSc KSPSetTolerances routine.

**Parameters:**

**KSP\_** The Ksp to set the tolerances for  
**RTOL** The relative tolerance to set  
**ATOL** The absolute tolerance to set  
**DTOL** The divergence tolerance to set  
**MAXITS** The maximum number of iterations  
**ERR** The error code  
**ERROR** The error string

Definition at line 1666 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### **6.9.2.28 subroutine CMISS\_PETSC::PETSC\_KSPSETTYPE (TYPE(PETSC\_KSP\_TYPE),intent(inout) KSP\_, METHOD, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Buffer routine to the PETSc KSPSetType routine.

**Parameters:**

**KSP\_** The Ksp to set the type for  
**ERR** The error code  
**ERROR** The error string

Definition at line 1700 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.9.2.29 subroutine CMISS\_PETSC::PETSC\_KSPSETUP (TYPE(PETSC\_KSP\_TYPE),intent(inout) *KSP\_*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc KSPSetUp routine.

**Parameters:**

***KSP\_*** The Ksp to set up

***ERR*** The error code

***ERROR*** The error string

Definition at line 1731 of file cmiss\_petsc.f90.

References   BASE\_ROUTINES::ENTERS(),   BASE\_ROUTINES::ERRORS(),   and   BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.30 subroutine CMISS\_PETSC::PETSC\_KSPSOLVE (TYPE(PETSC\_KSP\_TYPE),intent(inout) *KSP\_*, TYPE(PETSC\_VEC\_TYPE),intent(inout) *B*, TYPE(PETSC\_VEC\_TYPE),intent(inout) *X*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc KSPSolve routine.

**Parameters:**

***KSP\_*** The Ksp to set up

***B*** The RHS vector

***X*** The solution vector

***ERR*** The error code

***ERROR*** The error string

Definition at line 1761 of file cmiss\_petsc.f90.

References   BASE\_ROUTINES::ENTERS(),   BASE\_ROUTINES::ERRORS(),   and   BASE\_ROUTINES::EXITS().

Referenced by SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.9.2.31 subroutine CMISS\_PETSC::PETSC\_LOGPRINTSUMMARY (COMMUNICATOR, CHARACTER(LEN=\*),intent(in) *FILE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc PetscLogPrintSummary routine.

**Parameters:**

***FILE*** Filename for the log summary

***ERR*** The error code

***ERROR*** The error string

Definition at line 1793 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

#### 6.9.2.32 subroutine CMISS\_PETSC::PETSC\_MATASSEMBLYBEGIN (TYPE(PETSC\_MAT\_TYPE),intent(inout) *A*, ASSEMBLY\_TYPE, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Buffer routine to the PETSc MatAssemblyBegin routine.

##### Parameters:

***ERR*** The error code

***ERROR*** The error string

Definition at line 1824 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

#### 6.9.2.33 subroutine CMISS\_PETSC::PETSC\_MATASSEMBLYEND (TYPE(PETSC\_MAT\_TYPE),intent(inout) *A*, ASSEMBLY\_TYPE, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Buffer routine to the PETSc MatAssemblyEnd routine.

##### Parameters:

***A*** The matrix to assemble

***ERR*** The error code

***ERROR*** The error string

Definition at line 1855 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

#### 6.9.2.34 subroutine CMISS\_PETSC::PETSC\_MATCREATE (COMMUNICATOR, TYPE(PETSC\_MAT\_TYPE),intent(inout) *A*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \* ) [private]

Buffer routine to the PETSc MatCreate routine.

**Parameters:**

- A** On exit, the created matrix
- ERR** The error code
- ERROR** The error string

Definition at line 1886 of file cmiss\_petsc.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.35 subroutine CMISS\_PETSC::PETSC\_MATCREATEMPIAIJ (COMMUNICATOR, INTEGER(INTG),intent(in) LOCAL\_M, INTEGER(INTG),intent(in) LOCAL\_N, GLOB L\_M, INTEGER(INTG),intent(in) GLOBAL\_N, INTEGER(INTG),intent(in) DIAG\_NUMBER\_NZ\_PERROW, INTEGER(INTG),dimension(\*),intent(in) DIAG\_NUMBER\_NZ\_EACHROW, &,intent(in) OFFDIAG\_NUMBER\_NZ\_PERROW, INTEGER(INTG),dimension(\*),intent(in) OFFDIAG\_NUMBER\_NZ\_EACHROW, TYPE(PETSC\_MAT\_TYPE),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Buffer routine to the PETSc MatCreateMPIAIJ routine.

**Parameters:**

- LOCAL\_M** The number of local rows
- LOCAL\_N** The number of local columns
- GLOBAL\_N** The number of global columns
- DIAG\_NUMBER\_NZ\_PERROW** The maximum number of non-zeros per row in the diagonal part of the matrix
- DIAG\_NUMBER\_NZ\_EACHROW** The number of non-zeros per row in the diagonal part of the matrix
- OFFDIAG\_NUMBER\_NZ\_EACHROW** The number of non-zeros per row in the off-diagonal part of the matrix
- A** On exit, the matrix to create
- ERR** The error code
- ERROR** The error string

Definition at line 1917 of file cmiss\_petsc.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.36 subroutine CMISS\_PETSC::PETSC\_MATCREATEMPIDENSE (COMMUNICATOR, INTEGER(INTG),intent(in) LOCAL\_M, INTEGER(INTG),intent(in) LOCAL\_N, INTEGER(INTG),intent(in) GLOBAL\_M, INTEGER(INTG),intent(in) GLOBAL\_N, REAL(DP),dimension(\*),intent(in) MATRIX\_DATA, TYPE(PETSC\_MAT\_TYPE),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Buffer routine to the PETSc MatCreateMPIDense routine.

**Parameters:**

***LOCAL\_M*** The number of local rows  
***LOCAL\_N*** The number of local columns  
***GLOBAL\_M*** The number of global columns  
***GLOBAL\_N*** The number of global rows  
***MATRIX\_DATA*** Optional, the allocated matrix data.  
***A*** On exit, the matrix to create  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 1958 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.37 subroutine CMISS\_PETSC::PETSC\_MATCREATESEQAIJ (COMMUNICATOR, INTEGER(INTG),intent(in) *M*, INTEGER(INTG),intent(in) *N*, INTEGER(INTG),intent(in) *NUMBER\_NZ\_PERROW*, INTEGER(INTG),dimension(\*),intent(in) *NUMBER\_NZ\_EACHROW*, TYPE(PETSC\_MAT\_TYPE),intent(inout) *A*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc MatCreateSeqAIJ routine.

**Parameters:**

***M*** The number of rows  
***N*** The number of columns  
***NUMBER\_NZ\_PERROW*** The maximum number of non-zeros per row  
***NUMBER\_NZ\_EACHROW*** The number of non-zeros in each row  
***A*** On exit, the created matrix  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 1994 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.38 subroutine CMISS\_PETSC::PETSC\_MATCREATESEQDENSE (COMMUNICATOR, INTEGER(INTG),intent(in) *M*, INTEGER(INTG),intent(in) *N*, REAL(DP),dimension(\*),intent(in) *MATRIX\_DATA*, TYPE(PETSC\_MAT\_TYPE),intent(inout) *A*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc MatCreateSeqDense routine.

**Parameters:**

- M** The number of rows
- N** The number of columns
- MATRIX\_DATA** Optional, the allocated matrix data
- A** On exit, the created matrix
- ERR** The error code
- ERROR** The error string

Definition at line 2029 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

#### 6.9.2.39 subroutine CMISS\_PETSC::PETSC\_MATDESTROY (TYPE(PETSC\_- MAT\_TYPE),intent(inout) **A**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

Buffer routine to the PETSc MatDestroy routine.

**Parameters:**

- A** The matrix to destroy
- ERR** The error code
- ERROR** The error string

Definition at line 2063 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

#### 6.9.2.40 subroutine CMISS\_PETSC::PETSC\_MATFDCOLORINGCREATE (TYPE(PETSC\_- MAT\_TYPE),intent(inout) **A**, TYPE(PETSC\_ISCOLORING\_TYPE),intent(in) **ISCOLORING**, TYPE(PETSC\_MATFDCOLORING\_TYPE),intent(out) **FDCOLORING**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

Buffer routine to the PETSc MatFDColoringCreate routine.

**Parameters:**

- A** The PETSc matrix to create the FD coloring for
- ISCOLORING** The index set coloring to create the finite difference coloring for
- FDCOLORING** On exit, the matrix finite difference coloring
- ERR** The error code
- ERROR** The error string

Definition at line 2093 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.9.2.41 subroutine CMISS\_PETSC::PETSC\_MATFDCOLORINGDESTROY `(TYPE(PETSC_MATFDCOLORING_TYPE),intent(inout) MATFDCOLORING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Buffer routine to the PETSc MatFDColoringDestroy routine.

**Parameters:**

**MATFDCOLORING** The matrix finite difference coloring to destroy

**ERR** The error code

**ERROR** The error string

Definition at line 2125 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

#### 6.9.2.42 subroutine CMISS\_PETSC::PETSC\_MATFDCOLORINGFINALISE `(TYPE(PETSC_MATFDCOLORING_TYPE),intent(inout) MATFDCOLORING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

**Parameters:**

**MATFDCOLORING** The MatFDColoring to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 2155 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.9.2.43 subroutine CMISS\_PETSC::PETSC\_MATFDCOLORINGINITIALISE `(TYPE(PETSC_MATFDCOLORING_TYPE),intent(inout) MATFDCOLORING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

**Parameters:**

**MATFDCOLORING** The MatFDColoring to initialise

**ERR** The error code

**ERROR** The error string

Definition at line 2181 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESearch_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.9.2.44 subroutine CMISS\_PETSC::PETSC\_MATFDCOLORINGSETFROMOPTIONS `(TYPE(PETSC_MATFDCOLORING_TYPE),intent(inout) MATFDCOLORING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Buffer routine to the PETSc MatFDColoringSetFromOptions routine.

##### Parameters:

**MATFDCOLORING** The matrix finite difference coloring to set

**ERR** The error code

**ERROR** The error string

Definition at line 2205 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESearch_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.9.2.45 subroutine CMISS\_PETSC::PETSC\_MATFINALISE (TYPE(PETSC\_MAT\_TYPE),intent(inout) MAT\_, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)

##### Parameters:

**MAT\_** The MAT to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 2370 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

---

**6.9.2.46 subroutine CMISS\_PETSC::PETSC\_MATGETARRAY (TYPE(PETSC\_-  
MAT\_TYPE),intent(inout),target *A*, REAL(DP),dimension(:),pointer *ARRAY*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc MatGetArray routine.

**Parameters:**

- A* The matrix to get the array for
- ARRAY* On exit, a pointer to the matrix array
- ERR* The error code
- ERROR* The error string

Definition at line 2235 of file cmiss\_petsc.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.47 subroutine CMISS\_PETSC::PETSC\_MATGETCOLORING  
(TYPE(PETSC\_MAT\_TYPE),intent(inout) *A*, COLORING\_TYPE, TYPE(PETSC\_-  
ISCOLORING\_TYPE),intent(out) *ISCOLORING*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc MatGetColoring routine.

**Parameters:**

- A* The matrix to get the ownership range of
- ISCOLORING* On exit, the index set coloring
- ERR* The error code
- ERROR* The error string

Definition at line 2271 of file cmiss\_petsc.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Referenced by SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_CREATE\_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.9.2.48 subroutine CMISS\_PETSC::PETSC\_MATGETOWNERSHIPRANGE  
(TYPE(PETSC\_MAT\_TYPE),intent(inout) *A*, INTEGER(INTG),intent(out)  
*FIRST\_ROW*, INTEGER(INTG),intent(out) *LAST\_ROW*, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc MatGetOwnershipRange routine.

**Parameters:**

- A* The matrix to get the ownership range of

**FIRST\_ROW** On exit, the first row for the matrix

**LAST\_ROW** On exit, the last row for the matrix

**ERR** The error code

**ERROR** The error string

Definition at line 2303 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.49 subroutine CMISS\_PETSC::PETSC\_MATGETVALUES (TYPE(PETSC\_-  
MAT\_TYPE),intent(inout) A, INTEGER(INTG),intent(in) M,  
INTEGER(INTG),dimension(\*),intent(in) M\_INDICES, INTEGER(INTG),intent(in)  
N, INTEGER(INTG),dimension(\*),intent(in) N\_INDICES,  
REAL(DP),dimension(\*),intent(out) VALUES, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Buffer routine to the PETSc MatGetValues routine.

#### Parameters:

**A** The matrix to get the values of

**M** The number of row indices

**M\_INDICES** The row indices

**N** The number of column indices

**N\_INDICES** The column indices

**VALUES** The values to get

**ERR** The error code

**ERROR** The error string

Definition at line 2335 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.50 subroutine CMISS\_PETSC::PETSC\_MATINITIALISE (TYPE(PETSC\_-  
MAT\_TYPE),intent(inout) MAT\_, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

#### Parameters:

**MAT\_** The MAT to initialise

**ERR** The error code

**ERROR** The error string

Definition at line 2396 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

---

**6.9.2.51 subroutine CMISS\_PETSC::PETSC\_MATRESTOREARRAY  
 (TYPE(PETSC\_MAT\_TYPE),intent(inout) *A*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc MatRestoreArray routine.

**Parameters:**

- A*** The matrix to restore the array for
- ERR*** The error code
- ERROR*** The error string

Definition at line 2422 of file cmiss\_petsc.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.52 subroutine CMISS\_PETSC::PETSC\_MATSETLOCALTOGLOBALMAPPING  
 (TYPE(PETSC\_MAT\_TYPE),intent(inout) *A*, TYPE(PETSC\_-  
 ISLOCALTOGLOBALMAPPING\_TYPE),intent(in) *CTX*, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc MatSetLocalToGlobalMapping routine.

**Parameters:**

- A*** The matrix to set the local to global mapping for
- CTX*** The local to global mapping context
- ERR*** The error code
- ERROR*** The error string

Definition at line 2452 of file cmiss\_petsc.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.53 subroutine CMISS\_PETSC::PETSC\_MATSETOPTION (TYPE(PETSC\_-  
 MAT\_TYPE),intent(inout) *A*, *OPTION*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc MatSetOption routine.

**Parameters:**

- A*** The matrix to set the option for
- ERR*** The error code
- ERROR*** The error string

Definition at line 2483 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.54 subroutine CMISS\_PETSC::PETSC\_MATSETSIZES (TYPE(PETSC\_MAT\_TYPE),intent(inout) *A*, INTEGER(INTG),intent(in) *LOCAL\_M*, INTEGER(INTG),intent(in) *LOCAL\_N*, INTEGER(INTG),intent(in) *GLOBAL\_M*, INTEGER(INTG),intent(in) *GLOBAL\_N*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc MatSetSizes routine.

**Parameters:**

- A*** The matrix to set the size of
- LOCAL\_M*** Number of local rows
- LOCAL\_N*** Number of local columns
- GLOBAL\_M*** Number of global rows
- GLOBAL\_N*** Number of global columns
- ERR*** The error code
- ERROR*** The error string

Definition at line 2514 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.55 subroutine CMISS\_PETSC::PETSC\_MATSETVALUE (TYPE(PETSC\_MAT\_TYPE),intent(inout) *A*, INTEGER(INTG),intent(in) *ROW*, INTEGER(INTG),intent(in) *COL*, REAL(DP),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc MatSetValue routine.

**Parameters:**

- A*** The matrix to set the values of
- ROW*** The row index
- COL*** The column index
- VALUE*** The value to set
- ERR*** The error code
- ERROR*** The error string

Definition at line 2548 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

---

**6.9.2.56 subroutine CMISS\_PETSC::PETSC\_MATSETVALUELOCAL (TYPE(PETSC\_MAT\_-  
TYPE),intent(inout) *A*, INTEGER(INTG),intent(in) *ROW*, INTEGER(INTG),intent(in)  
*COL*, REAL(DP),intent(in) *VALUE*, INSERT\_MODE, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc MatSetValueLocal routine.

**Parameters:**

- A*** The matrix to set the values of
- ROW*** The row index
- COL*** The column index
- VALUE*** The value to set
- ERR*** The error code
- ERROR*** The error string

Definition at line 2618 of file cmiss\_petsc.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.57 subroutine CMISS\_PETSC::PETSC\_MATSETVALUES (TYPE(PETSC\_-  
MAT\_TYPE),intent(inout) *A*, INTEGER(INTG),intent(in)  
*M*, INTEGER(INTG),dimension(\*),intent(in) *M\_INDICES*,  
INTEGER(INTG),intent(in) *N*, INTEGER(INTG),dimension(\*),intent(in)  
*N\_INDICES*, REAL(DP),dimension(\*),intent(in) *VALUES*, INSERT\_MODE,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc MatSetValues routine.

**Parameters:**

- A*** The matrix to set the values of
- M*** The number of row indices
- M\_INDICES*** The row indices
- N*** The number of column indices
- N\_INDICES*** The column indices
- VALUES*** The values to set
- ERR*** The error code
- ERROR*** The error string

Definition at line 2582 of file cmiss\_petsc.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

---

**6.9.2.58 subroutine CMISS\_PETSC::PETSC\_MATSETVALUESLOCAL**  
 (TYPE(PETSC\_MAT\_TYPE),intent(inout) *A*, INTEGER(INTG),intent(in)  
*M*, INTEGER(INTG),dimension(\*),intent(in) *M\_INDICES*,  
 INTEGER(INTG),intent(in) *N*, INTEGER(INTG),dimension(\*),intent(in)  
*N\_INDICES*, REAL(DP),dimension(\*),intent(in) *VALUES*, INSERT\_MODE,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Buffer routine to the PETSc MatSetValuesLocal routine.

**Parameters:**

*A* The matrix to set the values of  
*M* The number of row indices  
*M\_INDICES* The row indices  
*N* The number of column indices  
*N\_INDICES* The column indices  
*VALUES* The values to set  
*ERR* The error code  
*ERROR* The error string

Definition at line 2652 of file cmiss\_petsc.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.59 subroutine CMISS\_PETSC::PETSC\_MATVIEW (TYPE(PETSC\_-MAT\_TYPE),intent(inout) *A*, *V*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc MatView routine.

**Parameters:**

*A* The matrix to view  
*ERR* The error code  
*ERROR* The error string

Definition at line 2688 of file cmiss\_petsc.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.60 subroutine CMISS\_PETSC::PETSC\_MATZEROENTRIES (TYPE(PETSC\_-MAT\_TYPE),intent(inout) *A*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc MatZeroEntries routine.

**Parameters:**

- A** The matrix to zero the entries of
- ERR** The error code
- ERROR** The error string

Definition at line 2719 of file cmiss\_petsc.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

#### 6.9.2.61 subroutine CMISS\_PETSC::PETSC\_PCFINALISE (TYPE(PETSC\_- PC\_TYPE),intent(inout) PC\_, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)

**Parameters:**

- PC\_** The PC to finalise
- ERR** The error code
- ERROR** The error string

Definition at line 2749 of file cmiss\_petsc.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.9.2.62 subroutine CMISS\_PETSC::PETSC\_PCINITIALISE (TYPE(PETSC\_- PC\_TYPE),intent(inout) PC\_, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)

**Parameters:**

- PC\_** The PC to initialise
- ERR** The error code
- ERROR** The error string

Definition at line 2775 of file cmiss\_petsc.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.9.2.63 subroutine CMISS\_PETSC::PETSC\_PCSETTYPE (TYPE(PETSC\_PC\_-  
TYPE),intent(inout) *PC\_*, METHOD, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc PCSetType routine.

**Parameters:**

*PC\_* The preconditioner to set the type of

*ERR* The error code

*ERROR* The error string

Definition at line 2799 of file cmiss\_petsc.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Referenced by SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_CREATE\_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.9.2.64 subroutine CMISS\_PETSC::PETSC\_SNESCREATE (COMMUNICATOR,  
TYPE(PETSC\_SNES\_TYPE),intent(inout) *SNES\_*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc SNESCreate routine.

**Parameters:**

*SNES\_* On exit, the SNES information

*ERR* The error code

*ERROR* The error string

Definition at line 2880 of file cmiss\_petsc.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Referenced by SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_CREATE\_FINISH(), and SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_TRUSTREGION\_CREATE\_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.9.2.65 subroutine CMISS\_PETSC::PETSC\_SNESDESTROY (TYPE(PETSC\_-  
SNES\_TYPE),intent(inout) *SNES\_*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Buffer routine to the PETSc SNESDestroy routine.

**Parameters:**

*SNES\_* The SNES to destroy

**ERR** The error code

**ERROR** The error string

Definition at line 2911 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

#### 6.9.2.66 subroutine CMISS\_PETSC::PETSC\_SNESFINALISE (TYPE(PETSC\_SNES\_TYPE),intent(inout) SNES\_, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

**Parameters:**

**SNES\_** The SNES to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 2830 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESearch_FINALISE()`, and `SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.9.2.67 subroutine CMISS\_PETSC::PETSC\_SNESGETCONVERGEDREASON (TYPE(PETSC\_SNES\_TYPE),intent(inout) SNES\_, INTEGER(INTG),intent(out) **REASON**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

Buffer routine to the PETSc SNESGetConvergedReason routine.

**Parameters:**

**SNES\_** The SNES to get the converged reason for

**REASON** On exit, the converged reason

**ERR** The error code

**ERROR** The error string

Definition at line 2941 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESearch_SOLVE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.9.2.68 subroutine CMISS\_PETSC::PETSC\_SNESGETFUNCTIONNORM (TYPE(PETSC\_SNES\_TYPE),intent(inout) SNES\_, REAL(DP),intent(out) FUNCTION\_NORM, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Buffer routine to the PETSc SNESGetFunctionNorm routine.

**Parameters:**

*SNES\_* The SNES to get the function norm for  
*FUNCTION\_NORM* On exit, the function norm  
*ERR* The error code  
*ERROR* The error string

Definition at line 2972 of file cmiss\_petsc.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.9.2.69 subroutine CMISS\_PETSC::PETSC\_SNESGETITERATIONNUMBER (TYPE(PETSC\_SNES\_TYPE),intent(inout) SNES\_, INTEGER(INTG),intent(out) ITERATION\_NUMBER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Buffer routine to the PETSc SNESGetIterationNumber routine.

**Parameters:**

*SNES\_* The SNES to get the iteration number for  
*ITERATION\_NUMBER* On exit, the number of iterations  
*ERR* The error code  
*ERROR* The error string

Definition at line 3003 of file cmiss\_petsc.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.9.2.70 subroutine CMISS\_PETSC::PETSC\_SNESINITIALISE (TYPE(PETSC\_SNES\_TYPE),intent(inout) SNES\_, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

**Parameters:**

*SNES\_* The snes to

**ERR** The error code

**ERROR** The error string

Definition at line 2856 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_INITIALISE()`, and `SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.9.2.71 subroutine CMISS\_PETSC::PETSC\_SNESLINESEARCHSET (TYPE(PETSC\_SNES\_TYPE),intent(inout) SNES\_, INTEGER(INTG),intent(in) LINESEARCH\_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)

Buffer routine to the PETSc SNESLineSearchSet routine.

##### Parameters:

**SNES\_** The SNES to set the line search for

**LINESEARCH\_TYPE** The line search type

**ERR** The error code

**ERROR** The error string

Definition at line 3034 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.9.2.72 subroutine CMISS\_PETSC::PETSC\_SNESLINESEARCHSETPARAMS (TYPE(PETSC\_SNES\_TYPE),intent(inout) SNES\_, REAL(DP),intent(in) ALPHA, REAL(DP),intent(in) MAXSTEP, REAL(DP),intent(in) STEPTOL, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)

Buffer routine to the PETSc SNESLineSearchSetParams routine.

##### Parameters:

**SNES\_** The SNES to set the line search parameters for

**ALPHA** The scalar such that  $0.5f_{n+1} \cdot f_{n+1} \leq .5*f_n \cdot f_n - \alpha |f_n \cdot J \cdot f_n|$

**MAXSTEP** The maximum norm of the update vector

**STEPTOL** the minimum norm fraction of the the original step after scaling

**ERR** The error code

**ERROR** The error string

Definition at line 3076 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.9.2.73 subroutine CMISS\_PETSC::PETSC\_SNESSETFROMOPTIONS `(TYPE(PETSC_SNES_TYPE),intent(inout) SNES_, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Buffer routine to the PETSc SNESSetFromOptions routine.

**Parameters:**

`SNES_` The SNES to set from the command line options

`ERR` The error code

`ERROR` The error string

Definition at line 3109 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_CREATE_FINISH()`, and `SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.9.2.74 subroutine CMISS\_PETSC::PETSC\_SNESSETFUNCTION `(TYPE(PETSC_SNES_TYPE),intent(inout) SNES_, TYPE(PETSC_VEC_TYPE),intent(inout) F, FFUNCTOR, TYPE(SOLUTION_TYPE),pointer CTX, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Buffer routine to the PETSc SNESSetFunction routine.

**Parameters:**

`SNES_` The SNES to set the function for

`F` The residual vector

`CTX` The solver data to pass to the function

`ERR` The error code

`ERROR` The error string

Definition at line 3139 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_CREATE_FINISH()`, and `SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

```
6.9.2.75 subroutine CMISS_PETSC::PETSC_SNESSETJACOBIAN_MATFDCOLORING
 (TYPE(PETSC_SNES_TYPE),intent(inout) SNES_, TYPE(PETSC_MAT_TYPE),intent(inout) A,
 TYPE(PETSC_MAT_TYPE),intent(inout) B, JFUNCTION,
 TYPE(PETSC_MATFDCOLORING_TYPE) CTX, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Buffer routine to the PETSc SNESSetJacobian routine for MatFDColoring contexts.

#### Parameters:

- SNES\_** The SNES to set the function for
- A** The Jacobian matrix
- B** The Jacobian preconditioning matrix
- CTX** The MatFDColoring data to pass to the function
- ERR** The error code
- ERROR** The error string

Definition at line 3173 of file cmiss\_petsc.f90.

References **BASE\_ROUTINES::ENTERS()**, **BASE\_ROUTINES::ERRORS()**, and **BASE\_ROUTINES::EXITS()**.

Here is the call graph for this function:

```
6.9.2.76 subroutine CMISS_PETSC::PETSC_SNESSETJACOBIAN_SOLVER
 (TYPE(PETSC_SNES_TYPE),intent(inout) SNES_, TYPE(PETSC_MAT_TYPE),intent(inout) A,
 TYPE(PETSC_MAT_TYPE),intent(inout) B, JFUNCTION,
 TYPE(SOLVER_TYPE),pointer CTX, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Buffer routine to the PETSc SNESSetJacobian routine for solver contexts.

#### Parameters:

- SNES\_** The SNES to set the function for
- A** The Jacobian matrix
- B** The Jacobian preconditioning matrix
- CTX** The solver data to pass to the function
- ERR** The error code
- ERROR** The error string

Definition at line 3207 of file cmiss\_petsc.f90.

References **BASE\_ROUTINES::ENTERS()**, **BASE\_ROUTINES::ERRORS()**, and **BASE\_ROUTINES::EXITS()**.

Here is the call graph for this function:

**6.9.2.77 subroutine CMISS\_PETSC::PETSC\_SNESSETTOLERANCES**  
 (**TYPE(PETSC\_SNES\_TYPE),intent(inout) SNES\_**, **REAL(DP),intent(in) ABSTOL**,  
**REAL(DP),intent(in) RTOL**, **REAL(DP),intent(in) STOL**, **INTEGER(INTG),intent(in) MAXIT**,  
**INTEGER(INTG),intent(in) MAXF**, **INTEGER(INTG),intent(out) ERR**,  
**TYPE(VARYING\_STRING),intent(out) ERROR**, \*)

Buffer routine to the PETSc SNESSetTolerances routine.

**Parameters:**

**SNES\_** The SNES to set the tolerances for  
**ABSTOL** The absolute convergence tolerance  
**RTOL** The relative convergence tolerance  
**STOL** The convergence tolerance for the change in the solution between steps  
**MAXIT** The maximum number of iterations  
**MAXF** The maximum number of function evaluations  
**ERR** The error code  
**ERROR** The error string

Definition at line 3241 of file cmiss\_petsc.f90.

References **BASE\_ROUTINES::ENTERS()**, **BASE\_ROUTINES::ERRORS()**, and **BASE\_ROUTINES::EXITS()**.

Referenced by **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_CREATE\_FINISH()**, and **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_TRUSTREGION\_CREATE\_FINISH()**.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.9.2.78 subroutine CMISS\_PETSC::PETSC\_SNESSETTRUSTREGIONTOLERANCE**  
 (**TYPE(PETSC\_SNES\_TYPE),intent(inout) SNES\_**, **REAL(DP),intent(in) TRTOL**,  
**INTEGER(INTG),intent(out) ERR**, **TYPE(VARYING\_STRING),intent(out) ERROR**, \*)

Buffer routine to the PETSc SNESSetTrustRegionTolerance routine.

**Parameters:**

**SNES\_** The SNES to set the tolerances for  
**TRTOL** The trust region tolerance  
**ERR** The error code  
**ERROR** The error string

Definition at line 3276 of file cmiss\_petsc.f90.

References **BASE\_ROUTINES::ENTERS()**, **BASE\_ROUTINES::ERRORS()**, and **BASE\_ROUTINES::EXITS()**.

Referenced by **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_TRUSTREGION\_CREATE\_FINISH()**.

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.9.2.79 subroutine CMISS\_PETSC::PETSC\_SNESSETTYPE (TYPE(PETSC\_SNES\_-  
TYPE),intent(inout) SNES\_, METHOD, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Buffer routine to the PETSc SNESSetType routine.

**Parameters:**

**SNES\_** The SNES to set the type for

**ERR** The error code

**ERROR** The error string

Definition at line 3307 of file cmiss\_petsc.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Referenced by SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_CREATE\_FINISH(), and SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_TRUSTREGION\_CREATE\_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.9.2.80 subroutine CMISS\_PETSC::PETSC\_SNESSOLVE (TYPE(PETSC\_SNES\_-  
TYPE),intent(inout) SNES\_, TYPE(PETSC\_VEC\_TYPE),intent(inout) B,  
TYPE(PETSC\_VEC\_TYPE),intent(inout) X, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Buffer routine to the PETSc SNESolve routine.

**Parameters:**

**SNES\_** The SNES to solve

**B** The constant part of the equation

**X** The solution vector

**ERR** The error code

**ERROR** The error string

Definition at line 3338 of file cmiss\_petsc.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Referenced by SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.9.2.81 subroutine CMISS\_PETSC::PETSC\_VECASSEMBLYBEGIN  
(TYPE(PETSC\_VEC\_TYPE),intent(inout) X, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Buffer routine to the PETSc VecAssemblyBegin routine.

**Parameters:**

*X* The vector to begin the assembly of

*ERR* The error code

*ERROR* The error string

Definition at line 3422 of file cmiss\_petsc.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

#### 6.9.2.82 subroutine CMISS\_PETSC::PETSC\_VECASSEMBLYEND (TYPE(PETSC\_VEC\_TYPE),intent(inout) *X*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Buffer routine to the PETSc VecAssemblyEnd routine.

**Parameters:**

*X* The vector to end the assembly of

*ERR* The error code

*ERROR* The error string

Definition at line 3452 of file cmiss\_petsc.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

#### 6.9.2.83 subroutine CMISS\_PETSC::PETSC\_VECCREATE (COMMUNICATOR, TYPE(PETSC\_VEC\_TYPE),intent(inout) *X*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Buffer routine to the PETSc VecCreate routine.

**Parameters:**

*X* On exit, the created vector

*ERR* The error code

*ERROR* The error string

Definition at line 3482 of file cmiss\_petsc.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

---

**6.9.2.84 subroutine CMISS\_PETSC::PETSC\_VECCREATEGHOST (COMMUNICATOR, INTEGER(INTG),intent(in) LOCAL\_SIZE, INTEGER(INTG),intent(in) GLOBAL\_SIZE, INTEGER(INTG),intent(in) NUMBER\_GHOST, INTEGER(INTG),dimension(\*),intent(in) GHOSTS, TYPE(PETSC\_VEC\_TYPE),intent(inout) X, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Buffer routine to the PETSc VecCreateGhost routine.

**Parameters:**

**LOCAL\_SIZE** The number of local elements  
**GLOBAL\_SIZE** The number of global elements  
**NUMBER\_GHOST** The number of ghost elements  
**GHOSTS** The global location of the each ghost element  
**X** On exit, the created vector  
**ERR** The error code  
**ERROR** The error string

Definition at line 3513 of file cmiss\_petsc.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.85 subroutine CMISS\_PETSC::PETSC\_VECCREATEGHOSTWITHARRAY (COMMUNICATOR, INTEGER(INTG),intent(in) LOCAL\_SIZE, INTEGER(INTG),intent(in) GLOBAL\_SIZE, INTEGER(INTG),intent(in) NUMBER\_GHOST, INTEGER(INTG),dimension(\*),intent(in) GHOSTS, REAL(DP),dimension(\*),intent(out) ARRAY, TYPE(PETSC\_VEC\_TYPE),intent(inout) X, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Buffer routine to the PETSc VecCreateGhostWithArray routine.

**Parameters:**

**LOCAL\_SIZE** The number of local elements  
**GLOBAL\_SIZE** The number of global elements  
**NUMBER\_GHOST** The number of ghost elements  
**GHOSTS** The global location of the each ghost element  
**ARRAY** The preallocated array of matrix data  
**X** On exit, the created vector  
**ERR** The error code  
**ERROR** The error string

Definition at line 3548 of file cmiss\_petsc.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

---

**6.9.2.86 subroutine CMISS\_PETSC::PETSC\_VECCREATEMPI (COMMUNICATOR,  
INTEGER(INTG),intent(in) LOCAL\_SIZE, INTEGER(INTG),intent(in) GLOBAL\_SIZE,  
TYPE(PETSC\_VEC\_TYPE),intent(inout) X, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Buffer routine to the PETSc VecCreateMPI routine.

**Parameters:**

*LOCAL\_SIZE* The number of local elements

*GLOBAL\_SIZE* The number of global elements

*X* On exit, the created vector

*ERR* The error code

*ERROR* The error string

Definition at line 3584 of file cmiss\_petsc.f90.

References   BASE\_ROUTINES::ENTERS(),   BASE\_ROUTINES::ERRORS(),   and   BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.87 subroutine CMISS\_PETSC::PETSC\_VECCREATEMPIWITHARRAY  
(COMMUNICATOR, INTEGER(INTG),intent(in) LOCAL\_SIZE,  
INTEGER(INTG),intent(in) GLOBAL\_SIZE, REAL(DP),dimension(\*),intent(out)  
ARRAY, TYPE(PETSC\_VEC\_TYPE),intent(inout) X, INTEGER(INTG),intent(out)  
ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Buffer routine to the PETSc VecCreateMPIWithArray routine.

**Parameters:**

*LOCAL\_SIZE* The number of local elements

*GLOBAL\_SIZE* The number of global elements

*ARRAY* The preallocated array for the vector data

*X* On exit, the created vector

*ERR* The error code

*ERROR* The error string

Definition at line 3617 of file cmiss\_petsc.f90.

References   BASE\_ROUTINES::ENTERS(),   BASE\_ROUTINES::ERRORS(),   and   BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.88 subroutine CMISS\_PETSC::PETSC\_VECCREATESEQ (COMMUNICATOR,  
INTEGER(INTG),intent(in) SIZE, TYPE(PETSC\_VEC\_TYPE),intent(inout) X,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Buffer routine to the PETSc VecCreateSeq routine.

**Parameters:*****SIZE*** The size of the vector***X*** On exit, the created vector***ERR*** The error code***ERROR*** The error string

Definition at line 3651 of file cmiss\_petsc.f90.

References **BASE\_ROUTINES::ENTERS()**, **BASE\_ROUTINES::ERRORS()**, and **BASE\_ROUTINES::EXITS()**.

Here is the call graph for this function:

**6.9.2.89 subroutine CMISS\_PETSC::PETSC\_VCCREATESEQWITHARRAY  
(COMMUNICATOR, INTEGER(INTG),intent(in) *SIZE*,  
REAL(DP),dimension(\*),intent(out) *ARRAY*, TYPE(PETSC\_VEC\_TYPE),intent(inout)  
*X*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
\*)**

Buffer routine to the PETSc VecCreateSeqWithArray routine.

**Parameters:*****SIZE*** The size of the vector***ARRAY*** The preallocated array for the vector data***X*** On exit, the created vector***ERR*** The error code***ERROR*** The error string

Definition at line 3683 of file cmiss\_petsc.f90.

References **BASE\_ROUTINES::ENTERS()**, **BASE\_ROUTINES::ERRORS()**, and **BASE\_ROUTINES::EXITS()**.

Here is the call graph for this function:

**6.9.2.90 subroutine CMISS\_PETSC::PETSC\_VECDESTROY (TYPE(PETSC\_-  
VEC\_TYPE),intent(inout) *X*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc VecDestroy routine.

**Parameters:*****X*** The vector to destroy***ERR*** The error code***ERROR*** The error string

Definition at line 3716 of file cmiss\_petsc.f90.

References **BASE\_ROUTINES::ENTERS()**, **BASE\_ROUTINES::ERRORS()**, and **BASE\_ROUTINES::EXITS()**.

Here is the call graph for this function:

---

**6.9.2.91 subroutine CMISS\_PETSC::PETSC\_VECDUPLICATE (TYPE(PETSC\_VEC\_-  
TYPE),intent(inout) *OLD*, TYPE(PETSC\_VEC\_TYPE),intent(out) *NEW*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc VecDuplicate routine.

**Parameters:**

***OLD*** The vector to duplicate  
***NEW*** On exit, the new duplicated vector  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 3746 of file cmiss\_petsc.f90.

References   BASE\_ROUTINES::ENTERS(),   BASE\_ROUTINES::ERRORS(),   and   BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.92 subroutine CMISS\_PETSC::PETSC\_VECFINALISE (TYPE(PETSC\_-  
VEC\_TYPE),intent(inout) *VEC\_*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

**Parameters:**

***VEC\_*** The Vec to finalise  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 3370 of file cmiss\_petsc.f90.

References   BASE\_ROUTINES::ENTERS(),   BASE\_ROUTINES::ERRORS(),   and   BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.93 subroutine CMISS\_PETSC::PETSC\_VECGETARRAY (TYPE(PETSC\_-  
VEC\_TYPE),intent(inout),target *X*, REAL(DP),dimension(:),pointer *ARRAY*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc VecGetArray routine.

**Parameters:**

***X*** The vector to get the array of  
***ARRAY*** On exit, a pointer to the array of the vector  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 3777 of file cmiss\_petsc.f90.

References   BASE\_ROUTINES::ENTERS(),   BASE\_ROUTINES::ERRORS(),   and   BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

---

**6.9.2.94 subroutine CMISS\_PETSC::PETSC\_VECGETARRAYF90 (TYPE(PETSC\_-  
VEC\_TYPE),intent(inout),target *X*, REAL(DP),dimension(:),pointer *ARRAY*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc VecGetArrayF90 routine.

**Parameters:**

*X* The vector to get the array of  
*ARRAY* On exit, a pointer to the array of the vector  
*ERR* The error code  
*ERROR* The error string

Definition at line 3813 of file cmiss\_petsc.f90.

References   BASE\_ROUTINES::ENTERS(),   BASE\_ROUTINES::ERRORS(),   and   BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.95 subroutine CMISS\_PETSC::PETSC\_VECGETLOCALSIZE (TYPE(PETSC\_-  
VEC\_TYPE),intent(inout) *X*, INTEGER(INTG),intent(out) *SIZE*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc VecGetLocalSize routine.

**Parameters:**

*X* The vector to get the local size of  
*SIZE* On exit, the local size of the vector  
*ERR* The error code  
*ERROR* The error string

Definition at line 3848 of file cmiss\_petsc.f90.

References   BASE\_ROUTINES::ENTERS(),   BASE\_ROUTINES::ERRORS(),   and   BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.96 subroutine CMISS\_PETSC::PETSC\_VECGETOWNERSHIPRANGE  
(TYPE(PETSC\_VEC\_TYPE),intent(inout) *X*, INTEGER(INTG),intent(out)  
*LOW*, INTEGER(INTG),intent(out) *HIGH*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc VecGetOwnershipRange routine.

**Parameters:**

*X* The vector to get the ownership range of  
*LOW* On exit, the low end of the range  
*HIGH* On exit, the high end of the range

**ERR** The error code

**ERROR** The error string

Definition at line 3879 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.97 subroutine CMISS\_PETSC::PETSC\_VECGETSIZE (TYPE(PETSC\_-  
VEC\_TYPE),intent(inout) X, INTEGER(INTG),intent(out) SIZE,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Buffer routine to the PETSc VecGetSize routine.

#### Parameters:

**X** The vector to get the size of

**SIZE** On exit, the size of the vector

**ERR** The error code

**ERROR** The error string

Definition at line 3911 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.98 subroutine CMISS\_PETSC::PETSC\_VECGETVALUES (TYPE(PETSC\_-  
VEC\_TYPE),intent(inout) X, INTEGER(INTG),intent(in)  
N, INTEGER(INTG),dimension(\*),intent(in) INDICES,  
REAL(DP),dimension(\*),intent(out) VALUES, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Buffer routine to the PETSc VecGetValues routine.

#### Parameters:

**X** The vector to set the values for

**N** The number of indicies to get

**INDICES** The indices to get

**VALUES** On return, the values at the specified indicies

**ERR** The error code

**ERROR** The error string

Definition at line 3942 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

---

**6.9.2.99 subroutine CMISS\_PETSC::PETSC\_VECHOSTGETLOCALFORM**  
 (TYPE(PETSC\_VEC\_TYPE),intent(inout) *G*, TYPE(PETSC\_VEC\_TYPE),intent(inout) *L*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Buffer routine to the PETSc VecGhostGetLocalForm routine.

**Parameters:**

- G* The global form of the vector
- L* On exit, the local form of the vector with ghosts
- ERR* The error code
- ERROR* The error string

Definition at line 3975 of file cmiss\_petsc.f90.

References   BASE\_ROUTINES::ENTERS(),   BASE\_ROUTINES::ERRORS(),   and   BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.100 subroutine CMISS\_PETSC::PETSC\_VECHOSTRESTORELOCALFORM**  
 (TYPE(PETSC\_VEC\_TYPE),intent(inout) *G*, TYPE(PETSC\_VEC\_TYPE),intent(inout) *L*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Buffer routine to the PETSc VecGhostRestoreLocalForm routine.

**Parameters:**

- G* The global form of the vector
- L* The local form of the vector
- ERR* The error code
- ERROR* The error string

Definition at line 4006 of file cmiss\_petsc.f90.

References   BASE\_ROUTINES::ENTERS(),   BASE\_ROUTINES::ERRORS(),   and   BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.101 subroutine CMISS\_PETSC::PETSC\_VECHOSTUPDATEBEGIN**  
 (TYPE(PETSC\_VEC\_TYPE),intent(inout) *X*, INSERT\_MODE, SCATTER\_MODE, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Buffer routine to the PETSc VecGhostUpdateBegin routine.

**Parameters:**

- X* The vector to begin the ghost update for
- ERR* The error code

**ERROR** The error string

Definition at line 4037 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.102 subroutine CMISS\_PETSC::PETSC\_VECHOSTUPDATEEND**  
`(TYPE(PETSC_VEC_TYPE),intent(inout) X, INSERT_MODE, SCATTER_MODE,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`  
`*)`

Buffer routine to the PETSc VecGhostUpdateEnd routine.

**Parameters:**

*X* The vector to end the ghost update for

*ERR* The error code

*ERROR* The error string

Definition at line 4069 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.103 subroutine CMISS\_PETSC::PETSC\_VECINITIALISE** `(TYPE(PETSC_VEC_TYPE),intent(inout) VEC_, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

**Parameters:**

*VEC\_* The Vec to initialise

*ERR* The error code

*ERROR* The error string

Definition at line 3396 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.104 subroutine CMISS\_PETSC::PETSC\_VECRESTOREARRAY**  
`(TYPE(PETSC_VEC_TYPE),intent(inout) X, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Buffer routine to the PETSc VecRestoreArray routine.

**Parameters:**

*X* The vector to restore the array of

**ERR** The error code

**ERROR** The error string

Definition at line 4101 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.105 subroutine CMISS\_PETSC::PETSC\_VECRESTOREARRAYF90**  
`(TYPE(PETSC_VEC_TYPE),intent(inout) X, REAL(DP),dimension(:),pointer ARRAY,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`  
`*)`

Buffer routine to the PETSc VecRestoreArrayF90 routine.

#### Parameters:

**X** The vector to restore the array of

**ARRAY** A pointer to the data to restore

**ERR** The error code

**ERROR** The error string

Definition at line 4131 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.106 subroutine CMISS\_PETSC::PETSC\_VECSET (TYPE(PETSC\_VEC\_TYPE),intent(inout) X, REAL(DP),intent(in) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Buffer routine to the PETSc VecSet routine.

#### Parameters:

**X** The vector to set the value of

**VALUE** The value to set

**ERR** The error code

**ERROR** The error string

Definition at line 4162 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

---

**6.9.2.107 subroutine CMISS\_PETSC::PETSC\_VECSETFROMOPTIONS**  
 (TYPE(PETSC\_VEC\_TYPE),intent(inout) *X*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Buffer routine to the PETSc VecSetFromOptions routine.

**Parameters:**

*X* The vector to set the options for

*ERR* The error code

*ERROR* The error string

Definition at line 4193 of file cmiss\_petsc.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.108 subroutine CMISS\_PETSC::PETSC\_VECSETLOCALTOGLOBALMAPPING**  
 (TYPE(PETSC\_VEC\_TYPE),intent(inout) *X*, TYPE(PETSC\_ISLOCALTOGLOBALMAPPING\_TYPE),intent(in) *CTX*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Buffer routine to the PETSc VecSetLocalToGlobalMapping routine.

**Parameters:**

*X* The vector to set the local to global mapping for

*CTX* The local to global mapping context

*ERR* The error code

*ERROR* The error string

Definition at line 4223 of file cmiss\_petsc.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.9.2.109 subroutine CMISS\_PETSC::PETSC\_VECSETSIZES** (TYPE(PETSC\_VEC\_TYPE),intent(inout) *X*, INTEGER(INTG),intent(in) *LOCAL\_SIZE*,  
 INTEGER(INTG),intent(in) *GLOBAL\_SIZE*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Buffer routine to the PETSc VecSetSizes routine.

**Parameters:**

*X* The vector to set the sizes of

*LOCAL\_SIZE* The number of local elements

*GLOBAL\_SIZE* The number of global elements

**ERR** The error code

**ERROR** The error string

Definition at line 4322 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.110 subroutine CMISS\_PETSC::PETSC\_VECSETVALUES**  
`(TYPE(PETSC_VEC_TYPE),intent(inout) X, INTEGER(INTG),intent(in) N,`  
`INTEGER(INTG),dimension(*),intent(in) INDICES, REAL(DP),dimension(*),intent(in)`  
`VALUES, INSERT_MODE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_-`  
`STRING),intent(out) ERROR, *)`

Buffer routine to the PETSc VecSetValues routine.

#### Parameters:

**X** The vector to set the values for

**N** The number of indices

**INDICES** The indices

**VALUES** The values to set

**ERR** The error code

**ERROR** The error string

Definition at line 4254 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.111 subroutine CMISS\_PETSC::PETSC\_VECSETVALUESLOCAL**  
`(TYPE(PETSC_VEC_TYPE),intent(inout) X, INTEGER(INTG),intent(in) N,`  
`INTEGER(INTG),dimension(*),intent(in) INDICES, REAL(DP),dimension(*),intent(in)`  
`VALUES, INSERT_MODE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_-`  
`STRING),intent(out) ERROR, *)`

Buffer routine to the PETSc VecSetValuesLocal routine.

#### Parameters:

**X** The vector to set the values of

**N** The number of indices

**INDICES** The local indices

**VALUES** The values to set

**ERR** The error code

**ERROR** The error string

Definition at line 4288 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.9.2.112 subroutine CMISS\_PETSC::PETSC\_VECVIEW (TYPE(PETSC\_-  
VEC\_TYPE),intent(inout) *X*, *V*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Buffer routine to the PETSc VecView routine.

**Parameters:**

*X* The vector to view

*ERR* The error code

*ERROR* The error string

Definition at line 4354 of file cmiss\_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

### 6.9.3 Variable Documentation

**6.9.3.1 LOGICAL,save CMISS\_PETSC::PETSC\_HANDLE\_ERROR**

Definition at line 183 of file cmiss\_petsc.f90.

**6.9.3.2 INTEGER(INTG),parameter CMISS\_PETSC::PETSC\_SNES\_LINESEARCH\_CUBIC =  
4**

Definition at line 177 of file cmiss\_petsc.f90.

Referenced by SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_CREATE\_FINISH().

**6.9.3.3 INTEGER(INTG),parameter CMISS\_PETSC::PETSC\_SNES\_LINESEARCH\_NO = 2**

Definition at line 175 of file cmiss\_petsc.f90.

Referenced by SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_CREATE\_FINISH().

**6.9.3.4 INTEGER(INTG),parameter CMISS\_PETSC::PETSC\_SNES\_LINESEARCH\_-  
NONORMS = 1**

Definition at line 174 of file cmiss\_petsc.f90.

Referenced by SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_CREATE\_FINISH().

**6.9.3.5 INTEGER(INTG),parameter CMISS\_PETSC::PETSC\_SNES\_LINESEARCH\_-  
QUADRATIC = 3**

Definition at line 176 of file cmiss\_petsc.f90.

Referenced by SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_CREATE\_FINISH().

## 6.10 CMISS\_PETSC\_TYPES Namespace Reference

This module contains types related to the PETSc library.

### Classes

- struct [PETSC\\_IS\\_TYPE](#)
- struct [PETSC\\_ISLOCALTOGLOBALMAPPING\\_TYPE](#)
- struct [PETSC\\_ISCOLORING\\_TYPE](#)
- struct [PETSC\\_KSP\\_TYPE](#)
- struct [PETSC\\_MAT\\_TYPE](#)
- struct [PETSC\\_MATFDCOLORING\\_TYPE](#)
- struct [PETSC\\_PC\\_TYPE](#)
- struct [PETSC\\_SNES\\_TYPE](#)
- struct [PETSC\\_VEC\\_TYPE](#)

### 6.10.1 Detailed Description

This module contains types related to the PETSc library.

## 6.11 COMP\_ENVIRONMENT Namespace Reference

This module contains all computational environment variables.

### Classes

- struct [CACHE\\_TYPE](#)  
*Contains information on a cache hierarchy.*
- struct [COMPUTATIONAL\\_NODE\\_TYPE](#)  
*Contains information on a computational node containing a number of processors.*
- struct [MPI\\_COMPUTATIONAL\\_NODE\\_TYPE](#)  
*Contains information on the MPI type to transfer information about a computational node.*
- struct [COMPUTATIONAL\\_ENVIRONMENT\\_TYPE](#)  
*Contains information on the computational environment the program is running in.*

### Functions

- subroutine [COMPUTATIONAL\\_NODE\\_FINALISE](#) (COMPUTATIONAL\_NODE, ERR, ERROR,\*)  
*Finalises the computational node data structures and deallocates all memory.*
- subroutine [COMPUTATIONAL\\_NODE\\_INITIALISE](#) (COMPUTATIONAL\_NODE, RANK, ERR, ERROR,\*)  
*Initialises the computational node data structures.*
- subroutine [COMPUTATIONAL\\_NODE\\_MPI\\_TYPE\\_FINALISE](#) (ERR, ERROR,\*)  
*Finalises the data structure containing the MPI type information for the [COMPUTATIONAL\\_NODE\\_TYPE](#).*
- subroutine [COMPUTATIONAL\\_NODE\\_MPI\\_TYPE\\_INITIALISE](#) (COMPUTATIONAL\_NODE, ERR, ERROR,\*)  
*Initialises the data structure containing the MPI type information for the [COMPUTATIONAL\\_NODE\\_TYPE](#).*
- subroutine [COMPUTATIONAL\\_ENVIRONMENT\\_FINALISE](#) (ERR, ERROR,\*)  
*Finalises the computational environment data structures and deallocates all memory.*
- subroutine [COMPUTATIONAL\\_ENVIRONMENT\\_INITIALISE](#) (ERR, ERROR,\*)  
*Initialises the computational environment data structures.*
- INTEGER(INTG) [COMPUTATIONAL\\_NODE\\_NUMBER\\_GET](#) (ERR, ERROR)  
*Returns the number/rank of the computational nodes.*
- INTEGER(INTG) [COMPUTATIONAL\\_NODES\\_NUMBER\\_GET](#) (ERR, ERROR)  
*Returns the number of computational nodes.*

## Variables

- TYPE(COMPUTATIONAL\_ENVIRONMENT\_TYPE) COMPUTATIONAL\_ENVIRONMENT  
*The computational environment the program is running in.*
- TYPE(MPI\_COMPUTATIONAL\_NODE\_TYPE) MPI\_COMPUTATIONAL\_NODE\_TYPE\_-  
**DATA**  
*The MPI data on the computational nodes.*

### 6.11.1 Detailed Description

This module contains all computational environment variables.

### 6.11.2 Function Documentation

#### 6.11.2.1 subroutine COMP\_ENVIRONMENT::COMPUTATIONAL\_ENVIRONMENT\_- FINALISE (INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

Finalises the computational environment data structures and deallocates all memory.

##### Parameters:

**ERR** The error code

**ERROR** The error string

Definition at line 278 of file computational\_environment.f90.

References COMPUTATIONAL\_ENVIRONMENT, COMPUTATIONAL\_NODE\_FINALISE(), COMPUTATIONAL\_NODE\_MPI\_TYPE\_FINALISE(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), CMISS\_MPI::MPI\_ERROR\_CHECK(), and CMISS\_PETSC::PETSC\_FINALIZE().

Referenced by CMISS::CMISS\_FINALISE(), and COMPUTATIONAL\_ENVIRONMENT\_-INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.11.2.2 subroutine COMP\_ENVIRONMENT::COMPUTATIONAL\_- ENVIRONMENT\_INITIALISE (INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*) [private]

Initialises the computational environment data structures.

##### Parameters:

**ERR** The error code

**ERROR** The error string

Definition at line 322 of file computational\_environment.f90.

References COMPUTATIONAL\_ENVIRONMENT, COMPUTATIONAL\_ENVIRONMENT\_FINALISE(), COMPUTATIONAL\_NODE\_INITIALISE(), COMPUTATIONAL\_NODE\_MPI\_TYPE\_INITIALISE(), BASE\_ROUTINES::DIAGNOSTIC\_OUTPUT\_TYPE, BASE\_ROUTINES::DIAGNOSTICS1, BASE\_ROUTINES::DIAGNOSTICS2, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), MPI\_COMPUTATIONAL\_NODE\_TYPE\_DATA, CMISS\_MPI::MPI\_ERROR\_CHECK(), and CMISS\_PETSC::PETSC\_INITIALIZE().

Referenced by CMISS::CMISS\_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.11.2.3 subroutine COMP\_ENVIRONMENT::COMPUTATIONAL\_NODE\_FINALISE**  
`(TYPE(COMPUTATIONAL_NODE_TYPE),intent(inout) COMPUTATIONAL_NODE,  
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`  
`[private]`

Finalises the computational node data structures and deallocates all memory.

#### Parameters:

**COMPUTATIONAL\_NODE** The computational node to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 116 of file computational\_environment.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by COMPUTATIONAL\_ENVIRONMENT\_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.11.2.4 subroutine COMP\_ENVIRONMENT::COMPUTATIONAL\_NODE\_INITIALISE**  
`(TYPE(COMPUTATIONAL_NODE_TYPE),intent(out) COMPUTATIONAL_NODE,  
 INTEGER(INTG),intent(in) RANK, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING_STRING),intent(out) ERROR, *)` [private]

Initialises the computational node data structures.

#### Parameters:

**COMPUTATIONAL\_NODE** The computational node to initialise

**RANK** The MPI rank of the computational node

**ERR** The error code

**ERROR** The error string

Definition at line 143 of file computational\_environment.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `CMISS_MPI::MPI_ERROR_CHECK()`.

Referenced by `COMPUTATIONAL_ENVIRONMENT_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### **6.11.2.5 subroutine `COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_TYPE_FINALISE` (`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *`) [private]**

Finalises the data structure containing the MPI type information for the [COMPUTATIONAL\\_NODE\\_TYPE](#).

**Parameters:**

***ERR*** The error code

***ERROR*** The error string

Definition at line 174 of file computational\_environment.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `MPI_COMPUTATIONAL_NODE_TYPE_DATA`, and `CMISS_MPI::MPI_ERROR_CHECK()`.

Referenced by `COMPUTATIONAL_ENVIRONMENT_FINALISE()`, and `COMPUTATIONAL_NODE_MPI_TYPE_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### **6.11.2.6 subroutine `COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_TYPE_INITIALISE` (`TYPE(COMPUTATIONAL_NODE_TYPE),intent(in) COMPUTATIONAL_NODE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *`) [private]**

Initialises the data structure containing the MPI type information for the [COMPUTATIONAL\\_NODE\\_TYPE](#).

**Parameters:**

***COMPUTATIONAL\_NODE*** The computational node containing the MPI type to initialise

***ERR*** The error code

***ERROR*** The error string

Definition at line 208 of file computational\_environment.f90.

References    `COMPUTATIONAL_NODE_MPI_TYPE_FINALISE()`,    `BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE`,    `BASE_ROUTINES::DIAGNOSTICS3`,    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `MPI_COMPUTATIONAL_NODE_TYPE_DATA`, and `CMISS_MPI::MPI_ERROR_CHECK()`.

Referenced by `COMPUTATIONAL_ENVIRONMENT_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

### 6.11.2.7 INTEGER(INTG) COMP\_ENVIRONMENT::COMPUTATIONAL\_NODE\_NUMBER\_- GET (INTEGER(INTG),intent(out) ***ERR***, TYPE(VARYING\_STRING),intent(out) ***ERROR***)

Returns the number/rank of the computational nodes.

**Parameters:**

***ERR*** The error code

***ERROR*** The error string

Definition at line 405 of file computational\_environment.f90.

References COMPUTATIONAL\_ENVIRONMENT, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by MESH\_ROUTINES::DECOMPOSITION\_ELEMENT\_DOMAIN\_CALCULATE(), MESH\_ROUTINES::DECOMPOSITION\_NUMBER\_OF\_DOMAINS\_SET(), DOMAIN\_MAPPINGS::DOMAIN\_MAPPINGS\_LOCAL\_FROM\_GLOBAL\_CALCULATE(), FIELD\_ROUTINES::FIELD\_INTERPOLATION\_PARAMETERS\_LINE\_GET(), FIELD\_IO\_ROUTINES::FIELD\_IO\_ELEMENTS\_EXPORT(), FIELD\_IO\_ROUTINES::FIELD\_IO\_FILEDS\_IMPORT(), and FIELD\_IO\_ROUTINES::FIELD\_IO\_NODES\_EXPORT().

Here is the call graph for this function:

Here is the caller graph for this function:

### 6.11.2.8 INTEGER(INTG) COMP\_ENVIRONMENT::COMPUTATIONAL\_- NODES\_NUMBER\_GET (INTEGER(INTG),intent(out) ***ERR***, TYPE(VARYING\_STRING),intent(out) ***ERROR***)

Returns the number of computational nodes.

**Parameters:**

***ERR*** The error code

***ERROR*** The error string

Definition at line 434 of file computational\_environment.f90.

References COMPUTATIONAL\_ENVIRONMENT, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by MESH\_ROUTINES::DECOMPOSITION\_ELEMENT\_DOMAIN\_CALCULATE(), MESH\_ROUTINES::DECOMPOSITION\_ELEMENT\_DOMAIN\_SET(), MESH\_ROUTINES::DECOMPOSITION\_NUMBER\_OF\_DOMAINS\_SET(), FIELD\_ROUTINES::FIELD\_INTERPOLATION\_PARAMETERS\_LINE\_GET(), FIELD\_IO\_ROUTINES::FIELD\_IO\_ELEMENTS\_EXPORT(), FIELD\_IO\_ROUTINES::FIELD\_IO\_FILEDS\_IMPORT(), and FIELD\_IO\_ROUTINES::FIELD\_IO\_NODES\_EXPORT().

Here is the call graph for this function:

Here is the caller graph for this function:

### 6.11.3 Variable Documentation

#### 6.11.3.1 TYPE(COMPUTATIONAL\_ENVIRONMENT\_TYPE) COMP\_ENVIRONMENT::COMPUTATIONAL\_ENVIRONMENT

The computational environment the program is running in.

Definition at line 97 of file computational\_environment.f90.

Referenced by CMISS::CMISS\_INITIALISE(), COMPUTATIONAL\_ENVIRONMENT\_FINALISE(), COMPUTATIONAL\_ENVIRONMENT\_INITIALISE(), COMPUTATIONAL\_NODE\_NUMBER\_GET(), COMPUTATIONAL\_NODES\_NUMBER\_GET(), MESH\_ROUTINES::DECOMPOSITION\_ELEMENT\_DOMAIN\_CALCULATE(), SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_CREATE\_FINISH(), SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESearch\_CREATE\_FINISH(), and SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_TRUSTREGION\_CREATE\_FINISH().

#### 6.11.3.2 TYPE(MPI\_COMPUTATIONAL\_NODE\_TYPE) COMP\_ENVIRONMENT::MPI\_COMPUTATIONAL\_NODE\_TYPE\_DATA

The MPI data on the computational nodes.

Definition at line 98 of file computational\_environment.f90.

Referenced by COMPUTATIONAL\_ENVIRONMENT\_INITIALISE(), COMPUTATIONAL\_NODE\_MPI\_TYPE\_FINALISE(), and COMPUTATIONAL\_NODE\_MPI\_TYPE\_INITIALISE().

## 6.12 COORDINATE\_ROUTINES Namespace Reference

This module contains all coordinate transformation and support routines.

### Classes

- struct [COORDINATE\\_SYSTEM\\_PTR\\_TYPE](#)
- struct [COORDINATE\\_SYSTEMS\\_TYPE](#)
- interface [COORDINATE\\_CONVERT\\_FROM\\_RC](#)
- interface [COORDINATE\\_CONVERT\\_TO\\_RC](#)
- interface [COORDINATE\\_DELTA\\_CALCULATE](#)
- interface [COORDINATE\\_SYSTEM\\_DIMENSION\\_SET](#)
- interface [COORDINATE\\_SYSTEM\\_FOCUS\\_SET](#)
- interface [COORDINATE\\_SYSTEM\\_RADIAL\\_INTERPOLATION\\_TYPE\\_SET](#)
- interface [COORDINATE\\_SYSTEM\\_TYPE\\_SET](#)
- interface [COORDINATE\\_SYSTEM\\_ORIGIN\\_SET](#)
- interface [COORDINATE\\_SYSTEM\\_ORIENTATION\\_SET](#)
- interface [DXZ](#)
- interface [D2ZX](#)
- interface [DZX](#)
- interface [COORDINATE\\_DERIVATIVE\\_CONVERT\\_TO\\_RC](#)
- interface [COORDINATE\\_SYSTEM\\_DESTROY](#)

### Functions

- REAL(DP) [COORDINATE\\_CONVERT\\_FROM\\_RC\\_DP](#) (COORDINATE\_SYSTEM, Z, ERR, ERROR)
- REAL(SP) [COORDINATE\\_CONVERT\\_FROM\\_RC\\_SP](#) (COORDINATE\_SYSTEM, Z, ERR, ER-ROR)
- REAL(DP) [COORDINATE\\_CONVERT\\_TO\\_RC\\_DP](#) (COORDINATE\_SYSTEM, X, ERR, ER-ROR)
- REAL(SP) [COORDINATE\\_CONVERT\\_TO\\_RC\\_SP](#) (COORDINATE\_SYSTEM, X, ERR, ER-ROR)
- REAL(DP) [COORDINATE\\_DELTA\\_CALCULATE\\_DP](#) (COORDINATE\_SYSTEM, X, Y, ERR, ERROR)
- subroutine [COORDINATE\\_METRICS\\_CALCULATE](#) (COORDINATE\_SYSTEM, JACOBIAN\_-TYPE, METRICS, ERR, ERROR,\*)
- subroutine [COORDINATE\\_SYSTEM\\_NORMAL\\_CALCULATE](#) (COORDINATE\_SYSTEM, RE-VERSE, X, N, ERR, ERROR,\*)
- INTEGER(INTG) [COORDINATE\\_SYSTEM\\_DIMENSION\\_GET](#) (COORDINATE\_SYSTEM)
- REAL(DP) [COORDINATE\\_SYSTEM\\_FOCUS\\_GET](#) (COORDINATE\_SYSTEM, ERR, ERROR)
- INTEGER(INTG) [COORDINATE\\_SYSTEM\\_RADIAL\\_INTERPOLATION\\_TYPE\\_GET](#) (COORDINATE\_SYSTEM)
- INTEGER(INTG) [COORDINATE\\_SYSTEM\\_TYPE\\_GET](#) (COORDINATE\_SYSTEM)
- subroutine [COORDINATE\\_SYSTEM\\_DIMENSION\\_SET\\_NUMBER](#) (USER\_NUMBER, DI-MENSION, ERR, ERROR,\*)
- subroutine [COORDINATE\\_SYSTEM\\_DIMENSION\\_SET\\_PTR](#) (COORDINATE\_SYSTEM, DI-MENSION, ERR, ERROR,\*)
- subroutine [COORDINATE\\_SYSTEM\\_FOCUS\\_SET\\_NUMBER](#) (USER\_NUMBER, FOCUS, ERR, ERROR,\*)

- subroutine `COORDINATE_SYSTEM_FOCUS_SET_PTR` (`COORDINATE_SYSTEM`, `FOCUS`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_NUMBER` (`USER_NUMBER`, `RADIAL_INTERPOLATION_TYPE`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR` (`COORDINATE_SYSTEM`, `RADIAL_INTERPOLATION_TYPE`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_TYPE_SET_NUMBER` (`USER_NUMBER`, `TYPE`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_TYPE_SET_PTR` (`COORDINATE_SYSTEM`, `TYPE`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_ORIGIN_SET_NUMBER` (`USER_NUMBER`, `ORIGIN`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_ORIGIN_SET_PTR` (`COORDINATE_SYSTEM`, `ORIGIN`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_ORIENTATION_SET_NUMBER` (`USER_NUMBER`, `ORIENTATION`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_ORIENTATION_SET_PTR` (`COORDINATE_SYSTEM`, `ORIENTATION`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_CREATE_START` (`USER_NUMBER`, `COORDINATE_SYSTEM`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_CREATE_FINISH` (`COORDINATE_SYSTEM`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_DESTROY_NUMBER` (`USER_NUMBER`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_DESTROY_PTR` (`COORDINATE_SYSTEM`, `ERR`, `ERROR,*`)
- REAL(DP) `DXZ_DP` (`COORDINATE_SYSTEM`, `I`, `X`, `ERR`, `ERROR`)
- REAL(DP) `D2ZX_DP` (`COORDINATE_SYSTEM`, `I`, `J`, `X`, `ERR`, `ERROR`)
- REAL(DP) `DZX_DP` (`COORDINATE_SYSTEM`, `I`, `X`, `ERR`, `ERROR`)
- subroutine `CO` (`COORDINATE_SYSTEM`, `PART_DERIV_TYPE`, `X`, `Z,&ERR`, `ERROR,*`)
- subroutine `CO` (`COORDINATE_SYSTEM`, `PART_DERIV_TYPE`, `X`, `Z,&ERR`, `ERROR,*`)
- subroutine `COORDINATE_DERIVATIVE_NORM` (`COORDINATE_SYSTEM`, `PART_DERIV_INDEX`, `INTERPOLATED_POINT`, `DERIV_NORM`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_INTERPOLATION_ADJUST` (`COORDINATE_SYSTEM`, `PARTIAL_DERIVATIVE_INDEX`, `VALUE`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_INTERPOLATION_PARAMETERS_ADJUST` (`COORDINATE_SYSTEM`, `INTERPOLATION_PARAMETERS`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_USER_NUMBER_FIND` (`USER_NUMBER`, `COORDINATE_SYSTEM`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEMS_FINALISE` (`ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEMS_INITIALISE` (`ERR`, `ERROR,*`)

## Variables

- INTEGER(INTG), parameter `COORDINATE_RECTANGULAR_CARTESIAN_TYPE` = 1  
*Rectangular Cartesian coordinate system type.*
- INTEGER(INTG), parameter `COORDINATE_CYCLINDRICAL_POLAR_TYPE` = 2  
*Cylindrical polar coordinate system type.*
- INTEGER(INTG), parameter `COORDINATE_SPHERICAL_POLAR_TYPE` = 3

*Spherical polar coordinate system type.*

- INTEGER(INTG), parameter **COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE** = 4  
*Prolate spheroidal coordinate system type.*
- INTEGER(INTG), parameter **COORDINATE\_OBLATE\_SPHEROIDAL\_TYPE** = 5  
*Oblate spheroidal coordinate system type.*
- INTEGER(INTG), parameter **COORDINATE\_NO\_RADIAL\_INTERPOLATION\_TYPE** = 0  
*No radial interpolation.*
- INTEGER(INTG), parameter **COORDINATE\_RADIAL\_INTERPOLATION\_TYPE** = 1  
*r radial interpolation*
- INTEGER(INTG), parameter **COORDINATE\_RADIAL\_SQUARED\_INTERPOLATION\_TYPE** = 2  
 *$r^2$  radial interpolation*
- INTEGER(INTG), parameter **COORDINATE\_RADIAL\_CUBED\_INTERPOLATION\_TYPE** = 3  
 *$r^3$  radial interpolation*
- INTEGER(INTG), parameter **COORDINATE\_JACOBIAN\_LINE\_TYPE** = 1  
*Line type Jacobian.*
- INTEGER(INTG), parameter **COORDINATE\_JACOBIAN\_AREA\_TYPE** = 2  
*Area type Jacobian.*
- INTEGER(INTG), parameter **COORDINATE\_JACOBIAN\_VOLUME\_TYPE** = 3  
*Volume type Jacobian.*
- CHARACTER(LEN=21) **COORDINAT**
- CHARACTER(LEN=21) **\_SYSTEM\_TYPE\_STRING** = & (/ , & , & , & , & /)
- TYPE(**COORDINATE\_SYSTEMS\_TYPE**) **COORDINATE\_SYSTEMS**
- TYPE(**COORDINATE\_SYSTEM\_TYPE**), pointer **GLOBAL\_COORDINATE\_SYSTEM**

### 6.12.1 Detailed Description

This module contains all coordinate transformation and support routines.

### 6.12.2 Function Documentation

#### 6.12.2.1 subroutine COORDINATE\_ROUTINES::CO (TYPE(COORDINATE\_SYSTEM\_TYPE),intent(in) **COORDINATE\_SYSTEM**, INTEGER(INTG),intent(in) **PART\_DERIV\_TYPE**, REAL(SP),dimension(:, :, :),intent(in) **X**, REAL(SP),dimension(:, :),intent(out) **Z**, &,intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \* ) [private]

Definition at line 3085 of file coordinate\_routines.f90.

References KINDS::SP, COORDINATE\_CYCLINDRICAL\_POLAR\_TYPE, COORDINATE\_OBLATE\_SPHEROIDAL\_TYPE, COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE, COORDINATE\_RECTANGULAR\_CARTESIAN\_TYPE, COORDINATE\_SPHERICAL\_POLAR\_TYPE, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.12.2.2 subroutine COORDINATE\_ROUTINES::CO (TYPE(COORDINATE\_SYSTEM\_TYPE),intent(in) COORDINATE\_SYSTEM, INTEGER(INTG),intent(in) PART\_DERIV\_TYPE, REAL(DP),dimension(:, :, :),intent(in) X, REAL(DP),dimension(:, :),intent(out) Z, &,intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 2516 of file coordinate\_routines.f90.

References KINDS::DP, COORDINATE\_CYCLINDRICAL\_POLAR\_TYPE, COORDINATE\_OBLATE\_SPHEROIDAL\_TYPE, COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE, COORDINATE\_RECTANGULAR\_CARTESIAN\_TYPE, COORDINATE\_SPHERICAL\_POLAR\_TYPE, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.12.2.3 REAL(DP) COORDINATE\_ROUTINES::COORDINATE\_CONVERT\_FROM\_RC\_DP (TYPE(COORDINATE\_SYSTEM\_TYPE),intent(in) COORDINATE\_SYSTEM, REAL(DP),dimension(:, :),intent(in) Z, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

Definition at line 227 of file coordinate\_routines.f90.

References KINDS::DP, COORDINATE\_CYCLINDRICAL\_POLAR\_TYPE, COORDINATE\_OBLATE\_SPHEROIDAL\_TYPE, COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE, COORDINATE\_RECTANGULAR\_CARTESIAN\_TYPE, COORDINATE\_SPHERICAL\_POLAR\_TYPE, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.12.2.4 REAL(SP) COORDINATE\_ROUTINES::COORDINATE\_CONVERT\_FROM\_RC\_SP (TYPE(COORDINATE\_SYSTEM\_TYPE),intent(in) COORDINATE\_SYSTEM, REAL(SP),dimension(:, :),intent(in) Z, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

Definition at line 352 of file coordinate\_routines.f90.

References KINDS::SP, COORDINATE\_CYCLINDRICAL\_POLAR\_TYPE, COORDINATE\_OBLATE\_SPHEROIDAL\_TYPE, COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE, COORDINATE\_RECTANGULAR\_CARTESIAN\_TYPE, COORDINATE\_SPHERICAL\_POLAR\_TYPE, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

---

**6.12.2.5 REAL(DP) COORDINATE\_ROUTINES::COORDINATE\_CONVERT\_TO\_RC\_DP**  
 (TYPE(COORDINATE\_SYSTEM\_TYPE),intent(in) *COORDINATE\_SYSTEM*,  
 REAL(DP),dimension(:),intent(in) *X*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]

Definition at line 488 of file coordinate\_routines.f90.

References KINDS::\_DP, COORDINATE\_CYCLINDRICAL\_POLAR\_TYPE, COORDINATE\_OBLATE\_SPHEROIDAL\_TYPE, COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE, COORDINATE\_RECTANGULAR\_CARTESIAN\_TYPE, COORDINATE\_SPHERICAL\_POLAR\_TYPE, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.12.2.6 REAL(SP) COORDINATE\_ROUTINES::COORDINATE\_CONVERT\_TO\_RC\_SP**  
 (TYPE(COORDINATE\_SYSTEM\_TYPE),intent(in) *COORDINATE\_SYSTEM*,  
 REAL(SP),dimension(:),intent(in) *X*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]

Definition at line 571 of file coordinate\_routines.f90.

References KINDS::\_SP, COORDINATE\_CYCLINDRICAL\_POLAR\_TYPE, COORDINATE\_OBLATE\_SPHEROIDAL\_TYPE, COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE, COORDINATE\_RECTANGULAR\_CARTESIAN\_TYPE, COORDINATE\_SPHERICAL\_POLAR\_TYPE, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.12.2.7 REAL(DP) COORDINATE\_ROUTINES::COORDINATE\_DELTA\_CALCULATE\_DP**  
 (TYPE(COORDINATE\_SYSTEM\_TYPE),intent(in) *COORDINATE\_SYSTEM*,  
 REAL(DP),dimension(:),intent(in) *X*, REAL(DP),dimension(:),intent(in) *Y*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*)  
 [private]

Definition at line 664 of file coordinate\_routines.f90.

References KINDS::\_DP, COORDINATE\_CYCLINDRICAL\_POLAR\_TYPE, COORDINATE\_OBLATE\_SPHEROIDAL\_TYPE, COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE, COORDINATE\_RECTANGULAR\_CARTESIAN\_TYPE, COORDINATE\_SPHERICAL\_POLAR\_TYPE, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.12.2.8 subroutine COORDINATE\_ROUTINES::COORDINATE\_DERIVATIVE\_NORM**  
 (TYPE(COORDINATE\_SYSTEM\_TYPE),pointer *COORDINATE\_SYSTEM*,  
 INTEGER(INTG),intent(in) *PART\_DERIV\_INDEX*, TYPE(FIELD\_-INTERPOLATED\_POINT\_TYPE),pointer *INTERPOLATED\_POINT*,  
 REAL(DP),intent(out) *DERIV\_NORM*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Definition at line 3654 of file coordinate\_routines.f90.

References KINDS::\_DP, COORDINATE\_CYCLINDRICAL\_POLAR\_TYPE, COORDINATE\_OBLATE\_SPHEROIDAL\_TYPE, COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE, COORDINATE\_-

RECTANGULAR\_CARTESIAN\_TYPE, COORDINATE\_SPHERICAL\_POLAR\_TYPE, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by FIELD\_ROUTINES::FIELD\_INTERPOLATION\_PARAMETERS\_LINE\_GET().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.12.2.9 subroutine COORDINATE\_ROUTINES::COORDINATE\_INTERPOLATION\_ADJUST (TYPE(COORDINATE\_SYSTEM\_TYPE),pointer COORDINATE\_SYSTEM, INTEGER(INTG),intent(in) PARTIAL\_DERIVATIVE\_INDEX, REAL(DP),intent(inout) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Definition at line 3752 of file coordinate\_routines.f90.

References KINDS::DP, COORDINATE\_CYCLINDRICAL\_POLAR\_TYPE, COORDINATE\_OBLATE\_SPHEROIDAL\_TYPE, COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE, COORDINATE\_RADIAL\_CUBED\_INTERPOLATION\_TYPE, COORDINATE\_RADIAL\_INTERPOLATION\_TYPE, COORDINATE\_RADIAL\_SQUARED\_INTERPOLATION\_TYPE, COORDINATE\_RECTANGULAR\_CARTESIAN\_TYPE, COORDINATE\_SPHERICAL\_POLAR\_TYPE, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by FIELD\_ROUTINES::FIELD\_INTERPOLATE\_GAUSS(), and FIELD\_ROUTINES::FIELD\_INTERPOLATE\_XI().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.12.2.10 subroutine COORDINATE\_ROUTINES::COORDINATE\_INTERPOLATION\_PARAMETERS\_ADJUST (TYPE(COORDINATE\_SYSTEM\_TYPE),pointer COORDINATE\_SYSTEM, TYPE(FIELD\_INTERPOLATION\_PARAMETERS\_TYPE),pointer INTERPOLATION\_PARAMETERS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Definition at line 3864 of file coordinate\_routines.f90.

References COORDINATE\_CYCLINDRICAL\_POLAR\_TYPE, COORDINATE\_OBLATE\_SPHEROIDAL\_TYPE, COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE, COORDINATE\_RADIAL\_CUBED\_INTERPOLATION\_TYPE, COORDINATE\_RADIAL\_INTERPOLATION\_TYPE, COORDINATE\_RADIAL\_SQUARED\_INTERPOLATION\_TYPE, COORDINATE\_RECTANGULAR\_CARTESIAN\_TYPE, COORDINATE\_SPHERICAL\_POLAR\_TYPE, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by FIELD\_ROUTINES::FIELD\_INTERPOLATION\_PARAMETERS\_ELEMENT\_GET(), and FIELD\_ROUTINES::FIELD\_INTERPOLATION\_PARAMETERS\_LINE\_GET().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.12.2.11 subroutine COORDINATE\_ROUTINES::COORDINATE\_METRICS\_CALCULATE**  
**(TYPE(COORDINATE\_SYSTEM\_TYPE),pointer COORDINATE\_SYSTEM,**  
**INTEGER(INTG),intent(in) JACOBIAN\_TYPE, TYPE(FIELD\_INTERPOLATED\_-**  
**POINT\_METRICS\_TYPE),pointer METRICS, INTEGER(INTG),intent(out) ERR,**  
**TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Definition at line 721 of file coordinate\_routines.f90.

References KINDS::\_DP, COORDINATE\_CYCLINDRICAL\_POLAR\_TYPE, COORDINATE\_JACOBIAN\_AREA\_TYPE, COORDINATE\_JACOBIAN\_LINE\_TYPE, COORDINATE\_JACOBIAN\_VOLUME\_TYPE, COORDINATE\_OBLATE\_SPHEROIDAL\_TYPE, COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE, COORDINATE\_RECTANGULAR\_CARTESIAN\_TYPE, COORDINATE\_SPHERICAL\_POLAR\_TYPE, BASE\_ROUTINES::DIAGNOSTIC\_OUTPUT\_TYPE, BASE\_ROUTINES::DIAGNOSTICS1, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and INPUT\_OUTPUT::WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES.

Referenced by FIELD\_ROUTINES::FIELD\_INTERPOLATED\_POINT\_METRICS\_CALCULATE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.12.2.12 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_CREATE\_-**  
**FINISH (TYPE(COORDINATE\_SYSTEM\_TYPE),pointer COORDINATE\_SYSTEM,**  
**INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,**  
**\*)**

Definition at line 1772 of file coordinate\_routines.f90.

References COORDINATE\_SYSTEMS, BASE\_ROUTINES::DIAGNOSTIC\_OUTPUT\_TYPE, BASE\_ROUTINES::DIAGNOSTICS1, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

**6.12.2.13 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_CREATE\_START**  
**(INTEGER(INTG),intent(in) USER\_NUMBER, TYPE(COORDINATE\_SYSTEM\_-**  
**TYPE),pointer COORDINATE\_SYSTEM, INTEGER(INTG),intent(out) ERR,**  
**TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Definition at line 1697 of file coordinate\_routines.f90.

References KINDS::\_DP, COORDINATE\_NO\_RADIAL\_INTERPOLATION\_TYPE, COORDINATE\_RECTANGULAR\_CARTESIAN\_TYPE, COORDINATE\_SYSTEM\_USER\_NUMBER\_FIND(), COORDINATE\_SYSTEMS, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

**6.12.2.14 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_DESTROY\_-**  
**NUMBER (INTEGER(INTG) USER\_NUMBER, INTEGER(INTG),intent(out) ERR,**  
**TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 1818 of file coordinate\_routines.f90.

References COORDINATE\_SYSTEMS, BASE\_ROUTINES::ENTERS(),  
ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

**6.12.2.15 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_DESTROY\_PTR  
(TYPE(COORDINATE\_SYSTEM\_TYPE),pointer COORDINATE\_SYSTEM,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*) [private]**

Definition at line 1874 of file coordinate\_routines.f90.

References COORDINATE\_SYSTEMS, BASE\_ROUTINES::ENTERS(),  
ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

**6.12.2.16 INTEGER(INTG) COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_-  
DIMENSION\_GET (TYPE(COORDINATE\_SYSTEM\_TYPE),intent(in)  
COORDINATE\_SYSTEM)**

Definition at line 1074 of file coordinate\_routines.f90.

**6.12.2.17 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_-  
DIMENSION\_SET\_NUMBER (INTEGER(INTG),intent(in) USER\_NUMBER,  
INTEGER(INTG),intent(in) DIMENSION, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 1180 of file coordinate\_routines.f90.

References COORDINATE\_SYSTEM\_DIMENSION\_SET\_PTR(), COORDINATE\_SYSTEM\_USER\_-  
NUMBER\_FIND(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-  
ROUTINES::EXITS().

Here is the call graph for this function:

**6.12.2.18 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_DIMENSION\_-  
SET\_PTR (TYPE(COORDINATE\_SYSTEM\_TYPE),pointer COORDINATE\_SYSTEM,  
INTEGER(INTG),intent(in) DIMENSION, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 1211 of file coordinate\_routines.f90.

References COORDINATE\_CYLINDRICAL\_POLAR\_TYPE, COORDINATE\_OBLATE\_-  
SPHEROIDAL\_TYPE, COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE, COORDINATE\_-  
RECTANGULAR\_CARTESIAN\_TYPE, COORDINATE\_SPHERICAL\_POLAR\_TYPE, BASE\_-  
ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by COORDINATE\_SYSTEM\_DIMENSION\_SET\_NUMBER().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.12.2.19 REAL(DP) COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_FOCUS\_GET  
(TYPE(COORDINATE\_SYSTEM\_TYPE),intent(in) COORDINATE\_SYSTEM,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR)**

Definition at line 1097 of file coordinate\_routines.f90.

References COORDINATE\_OBLATE\_SPHEROIDAL\_TYPE, COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.12.2.20 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_FOCUS\_SET\_-  
NUMBER (INTEGER(INTG),intent(in) USER\_NUMBER, REAL(DP),intent(in)  
FOCUS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out)  
ERROR, \*) [private]**

Definition at line 1281 of file coordinate\_routines.f90.

References COORDINATE\_SYSTEM\_FOCUS\_SET\_PTR(), COORDINATE\_SYSTEM\_USER\_-NUMBER\_FIND(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.12.2.21 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_-  
FOCUS\_SET\_PTR (TYPE(COORDINATE\_SYSTEM\_TYPE),pointer  
COORDINATE\_SYSTEM, REAL(DP),intent(in) FOCUS, INTEGER(INTG),intent(out)  
ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 1312 of file coordinate\_routines.f90.

References COORDINATE\_OBLATE\_SPHEROIDAL\_TYPE, COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by COORDINATE\_SYSTEM\_FOCUS\_SET\_NUMBER().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.12.2.22 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_-  
NORMAL\_CALCULATE (TYPE(COORDINATE\_SYSTEM\_TYPE),pointer  
COORDINATE\_SYSTEM, LOGICAL,intent(in) REVERSE,  
REAL(DP),dimension(:, :, intent(in) X, REAL(DP),dimension(3),intent(out) N,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*) [private]**

Definition at line 926 of file coordinate\_routines.f90.

References COORDINATE\_CYLINDRICAL\_POLAR\_TYPE, COORDINATE\_OBLATE\_SPHEROIDAL\_TYPE, COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE, COORDINATE\_RECTANGULAR\_CARTESIAN\_TYPE, COORDINATE\_SPHERICAL\_POLAR\_TYPE, BASE\_ROUTINES::DIAGNOSTIC\_OUTPUT\_TYPE, BASE\_ROUTINES::DIAGNOSTICS1, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.12.2.23 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_-ORIENTATION\_SET\_NUMBER (INTEGER(INTG),intent(in) *USER\_NUMBER*, REAL(DP),dimension(:,:),intent(in) *ORIENTATION*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Definition at line 1624 of file coordinate\_routines.f90.

References COORDINATE\_SYSTEM\_ORIENTATION\_SET\_PTR(), COORDINATE\_SYSTEM\_-USER\_NUMBER\_FIND(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.12.2.24 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_ORIENTATION\_SET\_PTR (TYPE(COORDINATE\_SYSTEM\_TYPE),pointer *COORDINATE\_SYSTEM*, REAL(DP),dimension(:,:),intent(in) *ORIENTATION*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Definition at line 1655 of file coordinate\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Referenced by COORDINATE\_SYSTEM\_ORIENTATION\_SET\_NUMBER().

Here is the call graph for this function:

**6.12.2.25 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_-ORIGIN\_SET\_NUMBER (INTEGER(INTG),intent(in) *USER\_NUMBER*, REAL(DP),dimension(:, intent(in) *ORIGIN*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Definition at line 1552 of file coordinate\_routines.f90.

References COORDINATE\_SYSTEM\_ORIGIN\_SET\_PTR(), COORDINATE\_SYSTEM\_USER\_-NUMBER\_FIND(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.12.2.26 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_ORIGIN\_SET\_PTR (TYPE(COORDINATE\_SYSTEM\_TYPE),pointer *COORDINATE\_SYSTEM*, REAL(DP),dimension(:, intent(in) *ORIGIN*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Definition at line 1583 of file coordinate\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Referenced by COORDINATE\_SYSTEM\_ORIGIN\_SET\_NUMBER().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.12.2.27 INTEGER(INTG) COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_-  
RADIAL\_INTERPOLATION\_TYPE\_GET (TYPE(COORDINATE\_SYSTEM\_-  
TYPE),intent(in) COORDINATE\_SYSTEM)**

Definition at line 1133 of file coordinate\_routines.f90.

**6.12.2.28 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_RADIAL\_-  
INTERPOLATION\_TYPE\_SET\_NUMBER (INTEGER(INTG),intent(in)  
USER\_NUMBER, INTEGER(INTG),intent(in) RADIAL\_INTERPOLATION\_TYPE,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*) [private]**

Definition at line 1364 of file coordinate\_routines.f90.

References COORDINATE\_SYSTEM\_RADIAL\_INTERPOLATION\_TYPE\_SET\_PTR(),  
COORDINATE\_SYSTEM\_USER\_NUMBER\_FIND(), BASE\_ROUTINES::ENTERS(), BASE\_-  
ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.12.2.29 subroutine COORDINATE\_ROUTINES::COORDINATE\_-  
SYSTEM\_RADIAL\_INTERPOLATION\_TYPE\_SET\_PTR  
(TYPE(COORDINATE\_SYSTEM\_TYPE),pointer COORDINATE\_SYSTEM,  
INTEGER(INTG),intent(in) RADIAL\_INTERPOLATION\_TYPE,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*) [private]**

Definition at line 1395 of file coordinate\_routines.f90.

References COORDINATE\_CYCLINDRICAL\_POLAR\_TYPE, COORDINATE\_NO\_RADIAL\_-  
INTERPOLATION\_TYPE, COORDINATE\_OBLATE\_SPHEROIDAL\_TYPE, COORDINATE\_-  
PROLATE\_SPHEROIDAL\_TYPE, COORDINATE\_RADIAL\_CUBED\_INTERPOLATION\_TYPE,  
COORDINATE\_RADIAL\_INTERPOLATION\_TYPE, COORDINATE\_RADIAL\_SQUARED\_-  
INTERPOLATION\_TYPE, COORDINATE\_RECTANGULAR\_CARTESIAN\_TYPE, COORDINATE\_-  
SPHERICAL\_POLAR\_TYPE, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and  
BASE\_ROUTINES::EXITS().

Referenced by COORDINATE\_SYSTEM\_RADIAL\_INTERPOLATION\_TYPE\_SET\_NUMBER().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.12.2.30 INTEGER(INTG) COORDINATE\_ROUTINES::COORDINATE\_-  
SYSTEM\_TYPE\_GET (TYPE(COORDINATE\_SYSTEM\_TYPE),intent(in)  
COORDINATE\_SYSTEM)**

Definition at line 1157 of file coordinate\_routines.f90.

---

**6.12.2.31 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_TYPE\_SET\_NUMBER (INTEGER(INTG),intent(in) *USER\_NUMBER*, INTEGER(INTG),intent(in) *TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Definition at line 1471 of file coordinate\_routines.f90.

References COORDINATE\_SYSTEM\_TYPE\_SET\_PTR(), COORDINATE\_SYSTEM\_USER\_NUMBER\_FIND(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

---

**6.12.2.32 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_TYPE\_SET\_PTR (TYPE(COORDINATE\_SYSTEM\_TYPE),pointer *COORDINATE\_SYSTEM*, INTEGER(INTG),intent(in) *TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Definition at line 1502 of file coordinate\_routines.f90.

References COORDINATE\_CYLINDRICAL\_POLAR\_TYPE, COORDINATE\_OBLATE\_SPHEROIDAL\_TYPE, COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE, COORDINATE\_RECTANGULAR\_CARTESIAN\_TYPE, COORDINATE\_SPHERICAL\_POLAR\_TYPE, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by COORDINATE\_SYSTEM\_TYPE\_SET\_NUMBER().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.12.2.33 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_USER\_NUMBER\_FIND (INTEGER(INTG),intent(in) *USER\_NUMBER*, TYPE(COORDINATE\_SYSTEM\_TYPE),pointer *COORDINATE\_SYSTEM*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Definition at line 3946 of file coordinate\_routines.f90.

References COORDINATE\_SYSTEMS, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), GLOBAL\_COORDINATE\_SYSTEM, and KINDS::PTR.

Referenced by COORDINATE\_SYSTEM\_CREATE\_START(), COORDINATE\_SYSTEM\_DIMENSION\_SET\_NUMBER(), COORDINATE\_SYSTEM\_FOCUS\_SET\_NUMBER(), COORDINATE\_SYSTEM\_ORIENTATION\_SET\_NUMBER(), COORDINATE\_SYSTEM\_ORIGIN\_SET\_NUMBER(), COORDINATE\_SYSTEM\_RADIAL\_INTERPOLATION\_TYPE\_SET\_NUMBER(), COORDINATE\_SYSTEM\_TYPE\_SET\_NUMBER(), REGION\_ROUTINES::REGION\_CREATE\_START(), and REGION\_ROUTINES::REGIONS\_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.12.2.34 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEMS\_FINALISE  
(INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
\*)**

Definition at line 3988 of file coordinate\_routines.f90.

References COORDINATE\_SYSTEMS, BASE\_ROUTINES::ENTERS(), BASE\_-ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), GLOBAL\_COORDINATE\_SYSTEM, and KINDS::PTR.

Referenced by CMISS::CMISS\_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.12.2.35 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEMS\_INITIALISE  
(INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
\*)**

Definition at line 4021 of file coordinate\_routines.f90.

References KINDS::\_DP, COORDINATE\_RECTANGULAR\_CARTESIAN\_TYPE, COORDINATE\_-SYSTEMS, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_-ROUTINES::EXITS(), GLOBAL\_COORDINATE\_SYSTEM, and KINDS::PTR.

Referenced by CMISS::CMISS\_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.12.2.36 REAL(DP) COORDINATE\_ROUTINES::D2ZX\_DP (TYPE(COORDINATE\_-  
SYSTEM\_TYPE),intent(in) *COORDINATE\_SYSTEM*, INTEGER(INTG),intent(in)  
*I*, INTEGER(INTG),intent(in) *J*, REAL(DP),dimension(:),intent(in) *X*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*)  
[private]**

Definition at line 2084 of file coordinate\_routines.f90.

References KINDS::\_DP, COORDINATE\_CYCLINDRICAL\_POLAR\_TYPE, COORDINATE\_-OBlate\_SPHEROIDAL\_TYPE, COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE, COORDINATE\_-RECTANGULAR\_CARTESIAN\_TYPE, COORDINATE\_SPHERICAL\_POLAR\_TYPE, BASE\_-ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.12.2.37 REAL(DP) COORDINATE\_ROUTINES::DXZ\_DP (TYPE(COORDINATE\_-  
SYSTEM\_TYPE),intent(in) *COORDINATE\_SYSTEM*, INTEGER(INTG),intent(in)  
*I*, REAL(DP),dimension(:),intent(in) *X*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]**

Definition at line 1943 of file coordinate\_routines.f90.

References KINDS::\_DP, COORDINATE\_CYCLINDRICAL\_POLAR\_TYPE, COORDINATE\_-OBlate\_SPHEROIDAL\_TYPE, COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE, COORDINATE\_-

RECTANGULAR\_CARTESIAN\_TYPE, COORDINATE\_SPHERICAL\_POLAR\_TYPE, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.12.2.38 REAL(DP) COORDINATE\_ROUTINES::DZX\_DP (TYPE(COORDINATE\_SYSTEM\_TYPE),intent(in) *COORDINATE\_SYSTEM*, INTEGER(INTG),intent(in) *I*, REAL(DP),dimension(:),intent(in) *X*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]**

Definition at line 2353 of file coordinate\_routines.f90.

References KINDS::\_DP, COORDINATE\_CYCLINDRICAL\_POLAR\_TYPE, COORDINATE\_OBLATE\_SPHEROIDAL\_TYPE, COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE, COORDINATE\_RECTANGULAR\_CARTESIAN\_TYPE, COORDINATE\_SPHERICAL\_POLAR\_TYPE, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

### 6.12.3 Variable Documentation

**6.12.3.1 CHARACTER(LEN=21) COORDINATE\_ROUTINES::\_SYSTEM\_TYPE\_STRING = & (/ , & , & , & , & /)**

Definition at line 106 of file coordinate\_routines.f90.

**6.12.3.2 CHARACTER(LEN=21) COORDINATE\_ROUTINES::COORDINAT**

Definition at line 106 of file coordinate\_routines.f90.

**6.12.3.3 TYPE(COORDINATE\_SYSTEMS\_TYPE) COORDINATE\_ROUTINES::COORDINATE\_SYSTEMS**

Definition at line 113 of file coordinate\_routines.f90.

Referenced by COORDINATE\_SYSTEM\_CREATE\_FINISH(), COORDINATE\_SYSTEM\_CREATE\_START(), COORDINATE\_SYSTEM\_DESTROY\_NUMBER(), COORDINATE\_SYSTEM\_DESTROY\_PTR(), COORDINATE\_SYSTEM\_USER\_NUMBER\_FIND(), COORDINATE\_SYSTEMS\_FINALISE(), and COORDINATE\_SYSTEMS\_INITIALISE().

**6.12.3.4 TYPE(COORDINATE\_SYSTEM\_TYPE),pointer COORDINATE\_ROUTINES::GLOBAL\_COORDINATE\_SYSTEM**

Definition at line 115 of file coordinate\_routines.f90.

Referenced by COORDINATE\_SYSTEM\_USER\_NUMBER\_FIND(), COORDINATE\_SYSTEMS\_FINALISE(), and COORDINATE\_SYSTEMS\_INITIALISE().

## 6.13 DOMAIN\_MAPPINGS Namespace Reference

This module handles all domain mappings routines.

### Functions

- subroutine `DOMAIN_MAPPINGS_ADJACENT_DOMAIN_FINALISE` (`ADJACENT_DOMAIN, ERR, ERROR,*`)  
*Finalises the adjacent domain and deallocates all memory for a domain mapping.*
- subroutine `DOMAIN_MAPPINGS_ADJACENT_DOMAIN_INITIALISE` (`ADJACENT_DOMAIN, ERR, ERROR,*`)  
*Initialise the adjacent domain for a domain mapping.*
- subroutine `DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE` (`DOMAIN_MAPPING, ERR, ERROR,*`)  
*Calculates the domain mappings local map from a domain mappings global map.*
- subroutine `DOMAIN_MAPPINGS_MAPPING_FINALISE` (`DOMAIN_MAPPING, ERR, ERROR,*`)  
*Finalises the mapping for a domain mappings mapping and deallocates all memory.*
- subroutine `DOMAIN_MAPPINGS_MAPPING_GLOBAL_FINALISE` (`MAPPING_GLOBAL_MAP, ERR, ERROR,*`)  
*Finalises the global mapping in the given domain mappings.*
- subroutine `DOMAIN_MAPPINGS_MAPPING_GLOBAL_INITIALISE` (`MAPPING_GLOBAL_MAP, ERR, ERROR,*`)  
*Finalises the global mapping in the given domain mappings.*
- subroutine `DOMAIN_MAPPINGS_MAPPING_INITIALISE` (`DOMAIN_MAPPING, NUMBER_OF_DOMAINS, ERR, ERROR,*`)  
*Initialises the mapping for a domain mappings mapping.*

### Variables

- INTEGER(INTG), parameter `DOMAIN_LOCAL_INTERNAL` = 1  
*The domain item is internal to the domain.*
- INTEGER(INTG), parameter `DOMAIN_LOCAL_BOUNDARY` = 2  
*The domain item is on the boundary of the domain.*
- INTEGER(INTG), parameter `DOMAIN_LOCAL_GHOST` = 3  
*The domain item is ghosted from another domain.*

#### 6.13.1 Detailed Description

This module handles all domain mappings routines.

### 6.13.2 Function Documentation

**6.13.2.1 subroutine DOMAIN\_MAPPINGS::DOMAIN\_MAPPINGS\_ADJACENT\_DOMAIN\_FINALISE (TYPE(DOMAIN\_ADJACENT\_DOMAIN\_TYPE) *ADJACENT\_DOMAIN*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Finalises the adjacent domain and deallocates all memory for a domain mapping.

**Parameters:**

***ADJACENT\_DOMAIN*** The adjacent domain to finalise.

***ERR*** The error code

***ERROR*** The error string

Definition at line 86 of file domain\_mappings.f90.

References **BASE\_ROUTINES::ENTERS()**, **BASE\_ROUTINES::ERRORS()**, and **BASE\_ROUTINES::EXITS()**.

Here is the call graph for this function:

**6.13.2.2 subroutine DOMAIN\_MAPPINGS::DOMAIN\_MAPPINGS\_ADJACENT\_DOMAIN\_INITIALISE (TYPE(DOMAIN\_ADJACENT\_DOMAIN\_TYPE) *ADJACENT\_DOMAIN*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Initialise the adjacent domain for a domain mapping.

**Parameters:**

***ADJACENT\_DOMAIN*** The adjacent domain to initialise

***ERR*** The error code

***ERROR*** The error string

Definition at line 114 of file domain\_mappings.f90.

References **BASE\_ROUTINES::ENTERS()**, **BASE\_ROUTINES::ERRORS()**, and **BASE\_ROUTINES::EXITS()**.

Here is the call graph for this function:

**6.13.2.3 subroutine DOMAIN\_MAPPINGS::DOMAIN\_MAPPINGS\_LOCAL\_FROM\_GLOBAL\_CALCULATE (TYPE(DOMAIN\_MAPPING\_TYPE),pointer *DOMAIN\_MAPPING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Calculates the domain mappings local map from a domain mappings global map.

**Parameters:**

***DOMAIN\_MAPPING*** The domain mapping to calculate the local mappings

***ERR*** The error code

**ERROR** The error string

Definition at line 140 of file domain\_mappings.f90.

References COMP\_ENVIRONMENT::COMPUTATIONAL\_NODE\_NUMBER\_GET(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), LISTS::LIST\_CREATE\_FINISH(), LISTS::LIST\_CREATE\_START(), LISTS::LIST\_DATA\_TYPE\_SET(), LISTS::LIST\_DESTROY(), LISTS::LIST\_INITIAL\_SIZE\_SET(), LISTS::LIST\_INTG\_TYPE, LISTS::LIST\_REMOVE\_DUPLICATES(), and KINDS::PTR.

Referenced by MESH\_ROUTINES::DECOMPOSITION\_NUMBER\_OF\_DOMAINS\_SET(), and FIELD\_ROUTINES::FIELD\_INTERPOLATION\_PARAMETERS\_LINE\_GET().

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.13.2.4 subroutine DOMAIN\_MAPPINGS::DOMAIN\_MAPPINGS\_MAPPING\_FINALISE (TYPE(DOMAIN\_MAPPING\_TYPE),pointer **DOMAIN\_MAPPING**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

Finalises the mapping for a domain mappings mapping and deallocates all memory.

**Parameters:**

**DOMAIN\_MAPPING** A pointer to the domain mapping to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 410 of file domain\_mappings.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by MESH\_ROUTINES::DECOMPOSITION\_NUMBER\_OF\_DOMAINS\_SET(), MESH\_ROUTINES::DOMAIN\_MAPPINGS\_NODES\_FINALISE(), EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_FINALISE(), and FIELD\_ROUTINES::FIELD\_INTERPOLATION\_PARAMETERS\_LINE\_GET().

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.13.2.5 subroutine DOMAIN\_MAPPINGS::DOMAIN\_MAPPINGS\_MAPPING\_GLOBAL\_FINALISE (TYPE(DOMAIN\_GLOBAL\_MAPPING\_TYPE) **MAPPING\_GLOBAL\_MAP**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*) [private]

Finalises the global mapping in the given domain mappings.

**Parameters:**

**MAPPING\_GLOBAL\_MAP** The domain global mapping to finalise

**ERR** The error code

**ERROR** Th error string

Definition at line 453 of file domain\_mappings.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.13.2.6 subroutine DOMAIN\_MAPPINGS::DOMAIN\_MAPPINGS\_MAPPING\_GLOBAL\_INITIALISE (TYPE(DOMAIN\_GLOBAL\_MAPPING\_TYPE)  
`MAPPING_GLOBAL_MAP`, INTEGER(INTG),intent(out) `ERR`,  
`TYPE(VARYING_STRING),intent(out) ERROR, *)`**

Finalises the global mapping in the given domain mappings.

**Parameters:**

`MAPPING_GLOBAL_MAP` The domain global mapping to initialise  
`ERR` The error code  
`ERROR` The error string

Definition at line 480 of file domain\_mappings.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET()`, and `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.13.2.7 subroutine DOMAIN\_MAPPINGS::DOMAIN\_MAPPINGS\_MAPPING\_INITIALISE (TYPE(DOMAIN\_MAPPING\_TYPE),pointer `DOMAIN_MAPPING`,  
`INTEGER(INTG),intent(in) NUMBER_OF_DOMAINS`, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`**

Initialises the mapping for a domain mappings mapping.

**Parameters:**

`DOMAIN_MAPPING` A pointer to the domain mapping to initialise the mappings for  
`NUMBER_OF_DOMAINS` The number of domains  
`ERR` The error code  
`ERROR` The error string

Definition at line 505 of file domain\_mappings.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET()`, `MESH_ROUTINES::DOMAIN_MAPPINGS_NODES_INITIALISE()`, and `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`.

Here is the call graph for this function:

Here is the caller graph for this function:

## 6.14 ELASTICITY\_ROUTINES Namespace Reference

This module handles all elasticity class routines.

### Functions

- subroutine `EL` (EQUATIONS\_SET, EQUATIONS\_TYPE, EQUATIONS\_SUBTYPE,&ERR, ERROR,\*)
 

*Sets/changes the problem type and subtype for an elasticity equation set class.*
- subroutine `ELASTICITYFINITEELEMENTCALCULATE` (EQUATIONS\_SET, ELEMENT\_NUMBER, ERR, ERROR,\*)
 

*Calculates the element stiffness matrices and rhs vector for the given element number for an elasticity class finite element equation set.*
- subroutine `ELASTICITYFINITEELEMENTJACOBIANEVALUATE` (EQUATIONS\_SET, ELEMENT\_NUMBER, ERR, ERROR,\*)
 

*Evaluates the Jacobian for the given element number for an elasticity class finite element equation set.*
- subroutine `ELASTICITYFINITEELEMENTRESIDUALCALCULATE` (EQUATIONS\_SET, ELEMENT\_NUMBER, ERR, ERROR,\*)
 

*Evaluates the residual and rhs vector for the given element number for an elasticity class finite element equation set.*
- subroutine `ELASTICITYEQUATIONSSETUP` (EQUATIONS\_SET, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)
 

*Sets up the equations set for an elasticity equations set class.*
- subroutine `ELASTICITYPROBLEMCLASSSETUP` (PROBLEM, PROBLEM\_EQUATION\_TYPE, PROBLEM\_SUBTYPE, ERR, ERROR,\*)
 

*Sets/changes the problem type and subtype for an elasticity problem class.*
- subroutine `ELASTICITYPROBLEMSSETUP` (PROBLEM, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)
 

*Sets up the problem for an elasticity problem class.*

### 6.14.1 Detailed Description

This module handles all elasticity class routines.

### 6.14.2 Function Documentation

- 6.14.2.1 subroutine `ELASTICITYROUTINES::EL` (TYPE(EQUATIONS\_SET\_TYPE),pointer EQUATIONS\_SET, INTEGER(INTG),intent(in) EQUATIONS\_TYPE, INTEGER(INTG),intent(in) EQUATIONS\_SUBTYPE, &,intent(out) ERR, INTEGER(INTG),intent(out) error, \*)

Sets/changes the problem type and subtype for an elasticity equation set class.

**Parameters:**

**EQUATIONS\_SET** A pointer to the equations set

**EQUATIONS\_TYPE** The equation type

**EQUATIONS\_SUBTYPE** The equation subtype

Definition at line 79 of file elasticity\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SETFINITE_ELASTICITY_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LINEAR_ELASTICITY_TYPE`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SUBTYPE_SET()`, and `LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SUBTYPE_SET()`.

Here is the call graph for this function:

```
6.14.2.2 subroutine ELASTICITY_ROUTINES::ELASTICITY_EQUATIONS_SET_SETUP (TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET,
INTEGER(INTG),intent(in) SETUP_TYPE, INTEGER(INTG),intent(in) ACTION_TYPE,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)
ERROR, *)
```

Sets up the equations set for an elasticity equations set class.

**Parameters:**

**EQUATIONS\_SET** A pointer to the equations set

**SETUP\_TYPE** The setup type

**ACTION\_TYPE** The action type

**ERR** The error code

**ERROR** The error string

Definition at line 237 of file elasticity\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SETFINITE_ELASTICITY_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LINEAR_ELASTICITY_TYPE`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SETUP()`, and `LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SETUP()`.

Here is the call graph for this function:

```
6.14.2.3 subroutine ELASTICITY_ROUTINES::ELASTICITY_FINITE_ELEMENT_CALCULATE (TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET,
INTEGER(INTG),intent(in) ELEMENT_NUMBER, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)
```

Calculates the element stiffness matrices and rhs vector for the given element number for an elasticity class finite element equation set.

**Parameters:**

**EQUATIONS\_SET** A pointer to the equations set

**ELEMENT\_NUMBER** The element number to calcualate

**ERR** The error code

**ERROR** The error string

Definition at line 120 of file elasticity\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SETFINITE_ELASTICITY_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LINEARELASTICITY_TYPE`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITYFINITE_ELEMENT_CALCULATE()`.

Here is the call graph for this function:

**6.14.2.4 subroutine ELASTICITY\_ROUTINES::ELASTICITYFINITE\_ELEMENT\_JACOBIAN\_EVALUATE (TYPE(EQUATIONS\_SET\_TYPE),pointer EQUATIONS\_SET, INTEGER(INTG),intent(in) ELEMENT\_NUMBER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Evaluates the Jacobian for the given element number for an elasticity class finite element equation set.

#### Parameters:

**EQUATIONS\_SET** A pointer to the equations set

**ELEMENT\_NUMBER** The element number to calcualate

**ERR** The error code

**ERROR** The error string

Definition at line 159 of file elasticity\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SETFINITE_ELASTICITY_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LINEARELASTICITY_TYPE`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITYFINITE_ELEMENT_JACOBIAN_EVALUATE()`.

Here is the call graph for this function:

**6.14.2.5 subroutine ELASTICITY\_ROUTINES::ELASTICITYFINITE\_ELEMENT\_RESIDUAL\_EVALUATE (TYPE(EQUATIONS\_SET\_TYPE),pointer EQUATIONS\_SET, INTEGER(INTG),intent(in) ELEMENT\_NUMBER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Evaluates the residual and rhs vector for the given element number for an elasticity class finite element equation set.

#### Parameters:

**EQUATIONS\_SET** A pointer to the equations set

**ELEMENT\_NUMBER** The element number to calcualate

**ERR** The error code

**ERROR** The error string

Definition at line 198 of file elasticity\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SETFINITE_ELASTICITY_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LINEAR_ELASTICITY_TYPE`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITYFINITE_ELEMENT_RESIDUAL_EVALUATE()`.

Here is the call graph for this function:

**6.14.2.6 subroutine `ELASTICITY_ROUTINES::ELASTICITY_PROBLEM_CLASS_TYPE_SET`** (`TYPE(PROBLEM_TYPE),pointer PROBLEM, INTEGER(INTG),intent(in) PROBLEM_EQUATION_TYPE, INTEGER(INTG),intent(in) PROBLEM_SUBTYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, */`)

Sets/changes the problem type and subtype for an elasticity problem class.

**Parameters:**

***PROBLEM*** A pointer to the problem

***PROBLEM\_EQUATION\_TYPE*** The problem type

***PROBLEM\_SUBTYPE*** The probobem subtype

***ERR*** The error code

***ERROR*** The error string

Definition at line 277 of file elasticity\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SETFINITE_ELASTICITY_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LINEAR_ELASTICITY_TYPE`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_PROBLEM_SUBTYPE_SET()`, and `LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SUBTYPE_SET()`.

Referenced by `PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_PTR()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.14.2.7 subroutine `ELASTICITY_ROUTINES::ELASTICITY_PROBLEM_SETUP`** (`TYPE(PROBLEM_TYPE),pointer PROBLEM, INTEGER(INTG),intent(in) SETUP_TYPE, INTEGER(INTG),intent(in) ACTION_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, */`)

Sets up the problem for an elasticity problem class.

**Parameters:**

***PROBLEM*** A pointer to the problem

***SETUP\_TYPE*** The setup type

***ACTION\_TYPE*** The action type

***ERR*** The error code

***ERROR*** The error string

Definition at line 317 of file elasticity\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `EQUATIONS_SET_CONSTANTS::EQUATIONS_SETFINITE_ELASTICITY_TYPE`,    `EQUATIONS_SET_CONSTANTS::EQUATIONS_SETLINEAR_ELASTICITY_TYPE`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_SETUP()`, and `LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_SETUP()`.

Referenced by `PROBLEM_ROUTINES::PROBLEM_SETUP()`.

Here is the call graph for this function:

Here is the caller graph for this function:

## 6.15 ELECTROMECHANICS\_ROUTINES Namespace Reference

This module handles all electromechanics class routines.

### 6.15.1 Detailed Description

This module handles all electromechanics class routines.

## 6.16 EQUATIONS\_MAPPING\_ROUTINES Namespace Reference

This module handles all equations mapping routines.

### Functions

- subroutine **EQUATIONS\_MAPPING\_CALCULATE** (EQUATIONS\_MAPPING, ERR, ERROR,\*)
 

*Calculates the equations/dofs mapping.*
- subroutine **EQUATIONS\_MAPPING\_CREATE\_FINISH** (EQUATIONS\_MAPPING, ERR, ERROR,\*)
 

*Finishes the process of creating an equations mapping.*
- subroutine **EQUATIONS\_MAPPING\_CREATE\_START** (EQUATIONS, EQUATIONS\_MAPPING, ERR, ERROR,\*)
 

*Finishes the process of creating an equations mapping for a problem solution.*
- subroutine **EQUATIONS\_MAPPING\_CREATE\_VALUES\_CACHE\_FINALISE** (CREATE\_VALUES\_CACHE, ERR, ERROR,\*)
 

*Finalises an equations mapping create values cache and deallocates all memory.*
- subroutine **EQUATIONS\_MAPPING\_CREATE\_VALUES\_CACHE\_INITIALISE** (EQUATIONS\_MAPPING, ERR, ERROR,\*)
 

*Initialises an equations mapping create values cache.*
- subroutine **EQUATIONS\_MAPPING\_DESTROY** (EQUATIONS\_MAPPING, ERR, ERROR,\*)
 

*Destroy an equations mapping.*
- subroutine **EQUATIONS\_MAPPING\_EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP\_FINALISE** (EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP, ERR, ERROR,\*)
 

*Finalises a variable to equations Jacobian map and deallocates all memory.*
- subroutine **EQUATIONS\_MAPPING\_EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP\_INITIALISE** (EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP, ERR, ERROR,\*)
 

*Initialises a variable to equations Jacobian map.*
- subroutine **EQUATIONS\_MAPPING\_EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_FINALISE** (EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP, ERR, ERROR,\*)
 

*Finalise an equations matrix to variable maps and deallocate all memory.*
- subroutine **EQUATIONS\_MAPPING\_EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_INITIALISE** (EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP, ERR, ERROR,\*)
 

*Initialise an equations matrix to variable maps.*
- subroutine **EQUATIONS\_MAPPING\_FINALISE** (EQUATIONS\_MAPPING, ERR, ERROR,\*)
 

*Finalises the equations mapping and deallocates all memory.*
- subroutine **EQUATIONS\_MAPPING\_INITIALISE** (EQUATIONS, ERR, ERROR,\*)
 

*Initialises the equations mapping and deallocates all memory.*

- subroutine **EQUATIONS\_MAPPING\_LINEAR\_MAPPING\_FINALISE** (LINEAR\_MAPPING, ERR, ERROR,\*)
 

*Finalises the equations mapping linear mapping and deallocates all memory.*
- subroutine **EQUATIONS\_MAPPING\_LINEAR\_MAPPING\_INITIALISE** (EQUATIONS\_MAPPING, ERR, ERROR,\*)
 

*Initialises the equations mapping linear mapping.*
- subroutine **EQUATIONS\_MAPPING\_LINEAR\_MATRICES\_COEFFICIENTS\_SET** (EQUATIONS\_MAPPING, MATRIX\_COEFFICIENTS, ERR, ERROR,\*)
 

*Sets the coefficients for the linear equations matrices in an equation set.*
- subroutine **EQUATIONS\_MAPPING\_LINEAR\_MATRICES\_NUMBER\_SET** (EQUATIONS\_MAPPING, NUMBER\_OF\_LINEAR\_EQUATIONS\_MATRICES, ERR, ERROR,\*)
 

*Sets the mapping between the dependent field variables and the linear equations matrices.*
- subroutine **EQUATIONS\_MAPPING\_LINEAR\_MATRICES\_VARIABLE\_TYPES\_SET** (EQUATIONS\_MAPPING, MATRIX\_VARIABLE\_TYPES, ERR, ERROR,\*)
 

*Sets the mapping between the dependent field variable types and the linear equations matrices.*
- subroutine **EQUATIONS\_MAPPING\_NONLINEAR\_MAPPING\_FINALISE** (NONLINEAR\_MAPPING, ERR, ERROR,\*)
 

*Finalises the equations mapping nonlinear mapping and deallocates all memory.*
- subroutine **EQUATIONS\_MAPPING\_NONLINEAR\_MAPPING\_INITIALISE** (EQUATIONS\_MAPPING, ERR, ERROR,\*)
 

*Initialises the equations mapping nonlinear mapping.*
- subroutine **EQUATIONS\_MAPPING\_RESIDUAL\_COEFFICIENT\_SET** (EQUATIONS\_MAPPING, RESIDUAL\_COEFFICIENT, ERR, ERROR,\*)
 

*Sets the coefficient applied to the equations set residual vector.*
- subroutine **EQUATIONS\_MAPPING\_RESIDUAL\_VARIABLE\_TYPE\_SET** (EQUATIONS\_MAPPING, RESIDUAL\_VARIABLE\_TYPE, ERR, ERROR,\*)
 

*Sets the mapping between a dependent field variable and the equations set residual vector.*
- subroutine **EQUATIONS\_MAPPING\_RHS\_COEFFICIENT\_SET** (EQUATIONS\_MAPPING, RHS\_COEFFICIENT, ERR, ERROR,\*)
 

*Sets the coefficient applied to the equations set RHS vector.*
- subroutine **EQUATIONS\_MAPPING\_RHS\_MAPPING\_FINALISE** (RHS\_MAPPING, ERR, ERROR,\*)
 

*Finalises the equations mapping RHS mapping and deallocates all memory.*
- subroutine **EQUATIONS\_MAPPING\_RHS\_MAPPING\_INITIALISE** (EQUATIONS\_MAPPING, ERR, ERROR,\*)
 

*Initialises the equations mapping RHS mapping.*
- subroutine **EQUATIONS\_MAPPING\_RHS\_VARIABLE\_TYPE\_SET** (EQUATIONS\_MAPPING, RHS\_VARIABLE\_TYPE, ERR, ERROR,\*)
 

*Sets the mapping between a dependent field variable and the equations set RHS vector.*

*Sets the mapping between a dependent field variable and the equations set rhs vector.*

- subroutine **EQUATIONS\_MAPPING\_SOURCE\_COEFFICIENT\_SET** (EQUATIONS\_MAPPING, SOURCE\_COEFFICIENT, ERR, ERROR,\*)

*Sets the coefficient applied to the equations set source vector.*

- subroutine **EQUATIONS\_MAPPING\_SOURCE\_MAPPING\_FINALISE** (SOURCE\_MAPPING, ERR, ERROR,\*)

*Finalises the equations mapping source mapping and deallocates all memory.*

- subroutine **EQUATIONS\_MAPPING\_SOURCE\_MAPPING\_INITIALISE** (EQUATIONS\_MAPPING, ERR, ERROR,\*)

*Initialises the equations mapping source mapping.*

- subroutine **EQUATIONS\_MAPPING\_SOURCE\_VARIABLE\_TYPE\_SET** (EQUATIONS\_MAPPING, SOURCE\_VARIABLE\_TYPE, ERR, ERROR,\*)

*Sets the mapping between a source field variable and the equations set source vector.*

- subroutine **EQUATIONS\_MAPPING\_VARIABLE\_TO\_EQUATIONS\_COLUMN\_MAP\_FINALISE** (VARIABLE\_TO\_EQUATIONS\_COLUMN\_MAP, ERR, ERROR,\*)

*Finalise an equations mapping equations matrix map.*

- subroutine **EQUATIONS\_MAPPING\_VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_MAP\_FINALISE** (VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_MAP, ERR, ERROR,\*)

*Finalises a variable to equations Jacobian map and deallocates all memory.*

- subroutine **EQUATIONS\_MAPPING\_VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_MAP\_INITIALISE** (VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_MAP, ERR, ERROR,\*)

*Initialises a variable to equations Jacobian map.*

- subroutine **EQUATIONS\_MAPPING\_VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP\_FINALISE** (VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP, ERR, ERROR,\*)

*Finalises a variable to equations matrices map and deallocates all memory.*

- subroutine **EQUATIONS\_MAPPING\_VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP\_INITIALISE** (VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP, ERR, ERROR,\*)

*Initialise an equations mapping equations matrix map.*

### 6.16.1 Detailed Description

This module handles all equations mapping routines.

### 6.16.2 Function Documentation

- 6.16.2.1 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_CALCULATE** (TYPE(EQUATIONS\_MAPPING\_TYPE),pointer **EQUATIONS\_MAPPING**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

Calculates the equations/dofs mapping.

**Parameters:**

**EQUATIONS\_MAPPING** A pointer to the equations mapping to calculate  
**ERR** The error code  
**ERROR** The error string

Definition at line 82 of file equations\_mapping\_routines.f90.

References            BASE\_ROUTINES::DIAGNOSTIC\_OUTPUT\_TYPE,            BASE\_-  
ROUTINES::DIAGNOSTICS1,            BASE\_ROUTINES::ENTERS(),            EQUATIONS\_SET\_-  
CONSTANTS::EQUATIONS\_SET\_LINEAR,            BASE\_ROUTINES::ERRORS(),            BASE\_-  
ROUTINES::EXITS(),            FIELD\_ROUTINES::FIELD\_NUMBER\_OF\_VARIABLE\_TYPES,            and  
KINDS::PTR.

Here is the call graph for this function:

**6.16.2.2 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_-  
CREATE\_FINISH (TYPE(EQUATIONS\_MAPPING\_TYPE),pointer  
EQUATIONS\_MAPPING, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Finishes the process of creating an equations mapping.

**Parameters:**

**EQUATIONS\_MAPPING** A pointer to the equations mapping  
**ERR** The error code  
**ERROR** The error string !<The error string

Definition at line 745 of file equations\_mapping\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-  
ROUTINES::EXITS().

Referenced by    FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITY\_EQUATIONS\_SET\_-  
SETUP(),    LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_EQUATIONS\_SET\_-  
STANDARD\_SETUP(),    and    LINEAR\_ELASTICITY\_ROUTINES::LINEAR\_ELASTICITY\_-  
EQUATIONS\_SET\_SETUP().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.16.2.3 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_-  
CREATE\_START (TYPE(EQUATIONS\_TYPE),pointer EQUATIONS,  
TYPE(EQUATIONS\_MAPPING\_TYPE),pointer EQUATIONS\_MAPPING,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)  
[private]**

Finishes the process of creating an equations mapping for a problem solution.

**Parameters:**

**EQUATIONS** A pointer to the equation to create the equations mapping from.  
**EQUATIONS\_MAPPING** On return, a pointer to the equations mapping. This must not be associated  
on entry

**ERR** The error code

**ERROR** The error string !<The error string

Definition at line 786 of file equations\_mapping\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by    `FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SETUP()`,    `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP()`,    and    `LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SETUP()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.16.2.4 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_CREATE\_VALUES\_CACHE\_FINALISE (TYPE(EQUATIONS\_MAPPING\_CREATE\_VALUES\_CACHE\_TYPE),pointer CREATE\_VALUES\_CACHE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Finalises an equations mapping create values cache and deallocates all memory.

#### Parameters:

**CREATE\_VALUES\_CACHE** A pointer to the create values cache

**ERR** The error code

**ERROR** The error string !<The error string

Definition at line 825 of file equations\_mapping\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.16.2.5 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_CREATE\_VALUES\_CACHE\_INITIALISE (TYPE(EQUATIONS\_MAPPING\_TYPE),pointer EQUATIONS\_MAPPING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Initialises an equations mapping create values cache.

#### Parameters:

**EQUATIONS\_MAPPING** A pointer to the create values cache

**ERR** The error code

**ERROR** The error string

Definition at line 853 of file equations\_mapping\_routines.f90.

References KINDS::DP, BASE\_ROUTINES::ENTERS(), EQUATIONS\_SET\_-CONSTANTS::EQUATIONS\_SET\_LINEAR, BASE\_ROUTINES::ERRORS(), BASE\_-ROUTINES::EXITS(), FIELD\_ROUTINES::FIELD\_NORMAL\_VARIABLE\_TYPE, FIELD\_-ROUTINES::FIELD\_STANDARD\_VARIABLE\_TYPE, and KINDS::PTR.

Here is the call graph for this function:

**6.16.2.6 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_-DESTROY (TYPE(EQUATIONS\_MAPPING\_TYPE),pointer *EQUATIONS\_MAPPING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Destroy an equations mapping.

**Parameters:**

***EQUATIONS\_MAPPING*** A pointer the equations mapping to destroy  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 1009 of file equations\_mapping\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.16.2.7 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_-MAPPING\_EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP\_FINALISE (TYPE(EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP\_TYPE)  
*EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \* ) [private]**

Finalises a variable to equations Jacobian map and deallocates all memory.

**Parameters:**

***EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP*** The equations Jacobian to variable map to finalise  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 1038 of file equations\_mapping\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.16.2.8 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_-MAPPING\_EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP\_INITIALISE (TYPE(EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP\_TYPE)  
*EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \* ) [private]**

Initialises a variable to equations Jacobian map.

**Parameters:**

***EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP*** The equations Jacobian to variable map to initialise

***ERR*** The error code

***ERROR*** The error string

Definition at line 1064 of file equations\_mapping\_routines.f90.

References **BASE\_ROUTINES::ENTERS()**, **BASE\_ROUTINES::ERRORS()**, and **BASE\_ROUTINES::EXITS()**.

Here is the call graph for this function:

**6.16.2.9 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_FINALISE (TYPE(EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_TYPE) EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Finalise an equations matrix to variable maps and deallocate all memory.

**Parameters:**

***EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP*** The equations matrix to variable map to finalise

***ERR*** The error code

***ERROR*** The error string

Definition at line 1094 of file equations\_mapping\_routines.f90.

References **BASE\_ROUTINES::ENTERS()**, **BASE\_ROUTINES::ERRORS()**, and **BASE\_ROUTINES::EXITS()**.

Here is the call graph for this function:

**6.16.2.10 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_INITIALISE (TYPE(EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_TYPE) EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Initialise an equations matrix to variable maps.

**Parameters:**

***EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP*** The equations matrix to variable map to initialise

***ERR*** The error code

***ERROR*** The error string

Definition at line 1120 of file equations\_mapping\_routines.f90.

References **KINDS::\_DP**, **BASE\_ROUTINES::ENTERS()**, **BASE\_ROUTINES::ERRORS()**, and **BASE\_ROUTINES::EXITS()**.

Here is the call graph for this function:

---

**6.16.2.11 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_FINALISE (TYPE(EQUATIONS\_MAPPING\_TYPE),pointer *EQUATIONS\_MAPPING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Finalises the equations mapping and deallocates all memory.

**Parameters:**

***EQUATIONS\_MAPPING*** A pointer to the equations mapping to finalise

***ERR*** The error code

***ERROR*** The error string

Definition at line 1150 of file equations\_mapping\_routines.f90.

References DOMAIN\_MAPPINGS::DOMAIN\_MAPPINGS\_MAPPING\_FINALISE(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

---

**6.16.2.12 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_INITIALISE (TYPE(EQUATIONS\_TYPE),pointer *EQUATIONS*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Initialises the equations mapping and deallocates all memory.

**Parameters:**

***EQUATIONS*** A pointer to the equations to initialise the equations mapping for

***ERR*** The error code

***ERROR*** The error string

Definition at line 1182 of file equations\_mapping\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

---

**6.16.2.13 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_LINEAR\_MAPPING\_FINALISE (TYPE(EQUATIONS\_MAPPING\_LINEAR\_TYPE),pointer *LINEAR\_MAPPING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Finalises the equations mapping linear mapping and deallocates all memory.

**Parameters:**

***LINEAR\_MAPPING*** A pointer to the linear mapping to finalise

***ERR*** The error code

***ERROR*** The error string

Definition at line 1226 of file equations\_mapping\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.16.2.14 subroutine EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_-
LINEAR_MAPPING_INITIALISE (TYPE(EQUATIONS_MAPPING_TYPE),pointer
EQUATIONS_MAPPING, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Initialises the equations mapping linear mapping.

**Parameters:**

***EQUATIONS\_MAPPING*** A pointer to the equations mapping to initialise the linear mapping for

***ERR*** The error code

***ERROR*** The error string

Definition at line 1269 of file equations\_mapping\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.16.2.15 subroutine EQUATIONS_MAPPING_ROUTINES::EQUATIONS_-
MAPPING_LINEAR_MATRICES_COEFFICIENTS_SET
(TYPE(EQUATIONS_MAPPING_TYPE),pointer EQUATIONS_MAPPING,
REAL(DP),dimension(:),intent(in) MATRIX_COEFFICIENTS,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
*)
```

Sets the coefficients for the linear equations matrices in an equation set.

**Parameters:**

***EQUATIONS\_MAPPING*** A pointer to the equations mapping.

***MATRIX\_COEFFICIENTS*** The matrix coefficients

***ERR*** The error code

***ERROR*** The error string

Definition at line 1308 of file equations\_mapping\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

---

**6.16.2.16 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_-  
LINEAR\_MATRICES\_NUMBER\_SET (TYPE(EQUATIONS\_MAPPING\_-  
TYPE),pointer *EQUATIONS\_MAPPING*, INTEGER(INTG),intent(in)  
*NUMBER\_OF\_LINEAR\_EQUATIONS\_MATRICES*, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Sets the mapping between the dependent field variables and the linear equations matrices.

**Parameters:**

***EQUATIONS\_MAPPING*** A pointer to the equations mapping to set the number of matrices for.  
***NUMBER\_OF\_LINEAR\_EQUATIONS\_MATRICES*** The number of linear equations matrices for  
the mapping.  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 1355 of file equations\_mapping\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_-  
SET\_LINEAR**,    **BASE\_ROUTINES::ERRORS()**,    **BASE\_ROUTINES::EXITS()**,    and    **FIELD\_-  
ROUTINES::FIELD\_NUMBER\_OF\_VARIABLE\_TYPES**.

Referenced by    **FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITY\_EQUATIONS\_SET\_-  
SETUP()**,    **LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_EQUATIONS\_SET\_-  
STANDARD\_SETUP()**,    and    **LINEAR\_ELASTICITY\_ROUTINES::LINEAR\_ELASTICITY\_-  
EQUATIONS\_SET\_SETUP()**.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.16.2.17 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_-  
MAPPING\_LINEAR\_MATRICES\_VARIABLE\_TYPES\_SET  
(TYPE(EQUATIONS\_MAPPING\_TYPE),pointer *EQUATIONS\_MAPPING*,  
INTEGER(INTG),dimension(:),intent(in) *MATRIX\_VARIABLE\_TYPES*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
\*)**

Sets the mapping between the dependent field variable types and the linear equations matrices.

**Parameters:**

***EQUATIONS\_MAPPING*** A pointer to the equations mapping  
***MATRIX\_VARIABLE\_TYPES*** The matrix variable types to map to each linear equations matrix  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 1476 of file equations\_mapping\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    **BASE\_-  
ROUTINES::EXITS()**,    **FIELD\_ROUTINES::FIELD\_NUMBER\_OF\_VARIABLE\_TYPES**,    and  
**KINDS::PTR**.

Referenced by    **LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_EQUATIONS\_-  
SET\_STANDARD\_SETUP()**,    and    **LINEAR\_ELASTICITY\_ROUTINES::LINEAR\_ELASTICITY\_-  
EQUATIONS\_SET\_SETUP()**.

Here is the call graph for this function:

Here is the caller graph for this function:

```
6.16.2.18 subroutine EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_-
NONLINEAR_MAPPING_FINALISE (TYPE(EQUATIONS_MAPPING_-
NONLINEAR_TYPE),pointer NONLINEAR_MAPPING, INTEGER(INTG),intent(out)-
ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Finalises the equations mapping nonlinear mapping and deallocates all memory.

**Parameters:**

**NONLINEAR\_MAPPING** A pointer to the nonlinear mapping to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 1572 of file equations\_mapping\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

```
6.16.2.19 subroutine EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_-
NONLINEAR_MAPPING_INITIALISE (TYPE(EQUATIONS_MAPPING_-
TYPE),pointer EQUATIONS_MAPPING, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Initialises the equations mapping nonlinear mapping.

**Parameters:**

**EQUATIONS\_MAPPING** A pointer to the equations mapping to initialise the nonlinear mapping for

**ERR** The error code

**ERROR** The error string

Definition at line 1604 of file equations\_mapping\_routines.f90.

References KINDS::\_DP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

```
6.16.2.20 subroutine EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_-
RESIDUAL_COEFFICIENT_SET (TYPE(EQUATIONS_MAPPING_TYPE),pointer
EQUATIONS_MAPPING, REAL(DP),intent(in) RESIDUAL_COEFFICIENT,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
*)
```

Sets the coefficient applied to the equations set residual vector.

**Parameters:**

**EQUATIONS\_MAPPING** A pointer to the equations mapping to set

**RESIDUAL\_COEFFICIENT** The coefficient applied to the equations set residual vector.

**ERR** The error code

**ERROR** The error string

Definition at line 1649 of file equations\_mapping\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.16.2.21 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_RESIDUAL\_VARIABLE\_TYPE\_SET** (`TYPE(EQUATIONS_MAPPING_TYPE),pointer EQUATIONS_MAPPING, INTEGER(INTG),intent(in) RESIDUAL_VARIABLE_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, */`)

Sets the mapping between a dependent field variable and the equations set residual vector.

#### Parameters:

**EQUATIONS\_MAPPING** A pointer to the equations mapping to set

**RESIDUAL\_VARIABLE\_TYPE** The variable type associated with the equations set residual vector.

**ERR** The error code

**ERROR** The error string

Definition at line 1690 of file equations\_mapping\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NONLINEAR`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `FIELD_ROUTINES::FIELD_NUMBER_OF_VARIABLE_TYPES`, and `KINDS::PTR`.

Here is the call graph for this function:

**6.16.2.22 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_RHS\_COEFFICIENT\_SET** (`TYPE(EQUATIONS_MAPPING_TYPE),pointer EQUATIONS_MAPPING, REAL(DP),intent(in) RHS_COEFFICIENT, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, */`)

Sets the coefficient applied to the equations set RHS vector.

#### Parameters:

**EQUATIONS\_MAPPING** A pointer to the equations mapping to set

**RHS\_COEFFICIENT** The coefficient applied to the equations set RHS vector.

**ERR** The error code

**ERROR** The error string

Definition at line 1773 of file equations\_mapping\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

---

**6.16.2.23 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_-  
RHS\_MAPPING\_FINALISE (TYPE(EQUATIONS\_MAPPING\_RHS\_TYPE),pointer  
RHS\_MAPPING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_-  
STRING),intent(out) ERROR, \*) [private]**

Finalises the equations mapping RHS mapping and deallocates all memory.

**Parameters:**

**RHS\_MAPPING** A pointer to the RHS mapping to finalise  
**ERR** The error code  
**ERROR** The error string

Definition at line 1814 of file equations\_mapping\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.16.2.24 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_-  
RHS\_MAPPING\_INITIALISE (TYPE(EQUATIONS\_MAPPING\_TYPE),pointer  
EQUATIONS\_MAPPING, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Initialises the equations mapping RHS mapping.

**Parameters:**

**EQUATIONS\_MAPPING** A pointer to the equations mapping to initialise the RHS mapping for  
**ERR** The error code  
**ERROR** The error string

Definition at line 1842 of file equations\_mapping\_routines.f90.

References KINDS::\_DP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.16.2.25 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_-  
RHS\_VARIABLE\_TYPE\_SET (TYPE(EQUATIONS\_MAPPING\_TYPE),pointer  
EQUATIONS\_MAPPING, INTEGER(INTG),intent(in) RHS\_VARIABLE\_TYPE,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*)**

Sets the mapping between a dependent field variable and the equations set rhs vector.

**Parameters:**

**EQUATIONS\_MAPPING** A pointer to the equations mapping to set  
**RHS\_VARIABLE\_TYPE** The variable type associated with the equations set rhs vector. If the problem does not have a rhs vector then the variable type on input should be zero.

**ERR** The error code

**ERROR** The error string

Definition at line 1883 of file equations\_mapping\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NONLINEAR`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_ROUTINES::FIELD_NUMBER_OF_VARIABLE_TYPES`, and `KINDS::PTR`.

Referenced by `FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SETUP()`, `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP()`, and `LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SETUP()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.16.2.26 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_SOURCE\_COEFFICIENT\_SET (TYPE(EQUATIONS\_MAPPING\_TYPE),pointer EQUATIONS\_MAPPING, REAL(DP),intent(in) SOURCE\_COEFFICIENT, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets the coefficient applied to the equations set source vector.

#### Parameters:

**EQUATIONS\_MAPPING** A pointer to the equations mapping to set

**SOURCE\_COEFFICIENT** The coefficient applied to the equations set source vector.

**ERR** The error code

**ERROR** The error string

Definition at line 1978 of file equations\_mapping\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.16.2.27 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_SOURCE\_MAPPING\_FINALISE (TYPE(EQUATIONS\_MAPPING\_SOURCE\_TYPE),pointer SOURCE\_MAPPING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Finalises the equations mapping source mapping and deallocates all memory.

#### Parameters:

**SOURCE\_MAPPING** A pointer to the SOURCE mapping to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 2019 of file equations\_mapping\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.16.2.28 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_SOURCE\_MAPPING\_INITIALISE (TYPE(EQUATIONS\_MAPPING\_TYPE),pointer EQUATIONS\_MAPPING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Initialises the equations mapping source mapping.

**Parameters:**

**EQUATIONS\_MAPPING** A pointer to the equations mapping to initialise the source mapping for  
**ERR** The error code  
**ERROR** The error string

Definition at line 2047 of file equations\_mapping\_routines.f90.

References `KINDS::_DP`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.16.2.29 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_SOURCE\_VARIABLE\_TYPE\_SET (TYPE(EQUATIONS\_MAPPING\_TYPE),pointer EQUATIONS\_MAPPING, INTEGER(INTG),intent(in) SOURCE\_VARIABLE\_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets the mapping between a source field variable and the equations set source vector.

**Parameters:**

**EQUATIONS\_MAPPING** A pointer to the equations mapping to set  
**SOURCE\_VARIABLE\_TYPE** The variable type associated with the equations set source vector. If the problem does not have a source vector then the variable type on input should be zero.  
**ERR** The error code  
**ERROR** The error string

Definition at line 2088 of file equations\_mapping\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_ROUTINES::FIELD_NUMBER_OF_VARIABLE_TYPES`, and `KINDS::PTR`.

Here is the call graph for this function:

---

**6.16.2.30 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_VARIABLE\_TO\_EQUATIONS\_COLUMN\_MAP\_FINALISE**  
`(TYPE(VARIABLE_TO_EQUATIONS_COLUMN_MAP_TYPE)`  
`VARIABLE_TO_EQUATIONS_COLUMN_MAP, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Finalise an equations mapping equations matrix map.

**Parameters:**

**VARIABLE\_TO\_EQUATIONS\_COLUMN\_MAP** The variable dof to equations column map to finalise  
**ERR** The error code  
**ERROR** The error string

Definition at line 2168 of file equations\_mapping\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.16.2.31 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_MAP\_FINALISE**  
`(TYPE(VARIABLE_TO_EQUATIONS_JACOBIAN_MAP_TYPE)`  
`VARIABLE_TO_EQUATIONS_JACOBIAN_MAP, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Finalises a variable to equations Jacobian map and deallocates all memory.

**Parameters:**

**VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_MAP** The variable to equations Jacobian map to finalise  
**ERR** The error code  
**ERROR** The error string

Definition at line 2194 of file equations\_mapping\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.16.2.32 subroutine EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_MAP\_INITIALISE**  
`(TYPE(VARIABLE_TO_EQUATIONS_JACOBIAN_MAP_TYPE)`  
`VARIABLE_TO_EQUATIONS_JACOBIAN_MAP, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Initialises a variable to equations Jacobian map.

**Parameters:**

**VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_MAP** The variable to equations Jacobian map to initialise

**ERR** The error code

**ERROR** The error string

Definition at line 2222 of file equations\_mapping\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.16.2.33 subroutine EQUATIONS_MAPPING_ROUTINES::EQUATIONS_-
 MAPPING_VARIABLE_TO_EQUATIONS_MATRICES_MAP_FINALISE
 (TYPE(VARIABLE_TO_EQUATIONS_MATRICES_MAP_TYPE)
 VARIABLE_TO_EQUATIONS_MATRICES_MAP, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Finalises a variable to equations matrices map and deallocates all memory.

#### Parameters:

**VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP** The variable to equations matrices map to initialise

**ERR** The error code

**ERROR** The error string

Definition at line 2248 of file equations\_mapping\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.16.2.34 subroutine EQUATIONS_MAPPING_ROUTINES::EQUATIONS_-
 MAPPING_VARIABLE_TO_EQUATIONS_MATRICES_MAP_INITIALISE
 (TYPE(VARIABLE_TO_EQUATIONS_MATRICES_MAP_TYPE)
 VARIABLE_TO_EQUATIONS_MATRICES_MAP, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Initialise an equations mapping equations matrix map.

#### Parameters:

**VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP** The variable to equations matrices map to initialise

**ERR** The error code

**ERROR** The error string

Definition at line 2284 of file equations\_mapping\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

## 6.17 EQUATIONS\_SET\_CONSTANTS Namespace Reference

This module defines all constants shared across equations set routines.

### Variables

- INTEGER(INTG), parameter `EQUATIONS_SET_NO_CLASS` = 0
- INTEGER(INTG), parameter `EQUATIONS_SET_ELASTICITY_CLASS` = 1
- INTEGER(INTG), parameter `EQUATIONS_SET_FLUID_MECHANICS_CLASS` = 2
- INTEGER(INTG), parameter `EQUATIONS_SET_ELECTROMAGNETICS_CLASS` = 3
- INTEGER(INTG), parameter `EQUATIONS_SET_CLASSICAL_FIELD_CLASS` = 4
- INTEGER(INTG), parameter `EQUATIONS_SET_MODAL_CLASS` = 5
- INTEGER(INTG), parameter `EQUATIONS_SET_FITTING_CLASS` = 6
- INTEGER(INTG), parameter `EQUATIONS_SET_OPTIMISATION_CLASS` = 7
- INTEGER(INTG), parameter `EQUATIONS_SET_NO_TYPE` = 0
- INTEGER(INTG), parameter `EQUATIONS_SET_LINEAR_ELASTICITY_TYPE` = 1
- INTEGER(INTG), parameter `EQUATIONS_SETFINITE_ELASTICITY_TYPE` = 2
- INTEGER(INTG), parameter `EQUATIONS_SET_STOKES_FLUID_TYPE` = 1
- INTEGER(INTG), parameter `EQUATIONS_SET_NAVIER_STOKES_FLUID_TYPE` = 2
- INTEGER(INTG), parameter `EQUATIONS_SET_ELECTROSTATIC_TYPE` = 1
- INTEGER(INTG), parameter `EQUATIONS_SET_MAGNETOSTATIC_TYPE` = 2
- INTEGER(INTG), parameter `EQUATIONS_SET_MAXWELLS_EQUATIONS_TYPE` = 3
- INTEGER(INTG), parameter `EQUATIONS_SET_LAPLACE_EQUATION_TYPE` = 1
- INTEGER(INTG), parameter `EQUATIONS_SET_POISSON_EQUATION_TYPE` = 2
- INTEGER(INTG), parameter `EQUATIONS_SET_HELMHOLTZ_EQUATION_TYPE` = 3
- INTEGER(INTG), parameter `EQUATIONS_SET_WAVE_EQUATION_TYPE` = 4
- INTEGER(INTG), parameter `EQUATIONS_SET_DIFFUSION_EQUATION_TYPE` = 5
- INTEGER(INTG), parameter `EQUATIONS_SET_ADVECTION_DIFFUSION_EQUATION_TYPE` = 6
- INTEGER(INTG), parameter `EQUATIONS_SETREACTION_DIFFUSION_EQUATION_TYPE` = 7
- INTEGER(INTG), parameter `EQUATIONS_SET_BIHAMMORPHIC_EQUATION_TYPE` = 8
- INTEGER(INTG), parameter `EQUATIONS_SET_LINEAR_ELASTIC_MODAL_TYPE` = 1
- INTEGER(INTG), parameter `EQUATIONS_SET_NO_SUBTYPE` = 0
- INTEGER(INTG), parameter `EQUATIONS_SET_STANDARD_LAPLACE_SUBTYPE` = 1
- INTEGER(INTG), parameter `EQUATIONS_SET_GENERALISED_LAPLACE_SUBTYPE` = 2
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANT_SOURCE_POISSON_SUBTYPE` = 1
- INTEGER(INTG), parameter `EQUATIONS_SET_LINEAR_SOURCE_POISSON_SUBTYPE` = 2
- INTEGER(INTG), parameter `EQUATIONS_SET_QUADRATIC_SOURCE_POISSON_SUBTYPE` = 2
- INTEGER(INTG), parameter `EQUATIONS_SET_SETUP_INITIAL_TYPE` = 1

*Initial setup.*

- INTEGER(INTG), parameter `EQUATIONS_SET_SETUP_GEOMETRY_TYPE` = 2

*Geometry setup.*

- INTEGER(INTG), parameter `EQUATIONS_SET_SETUP_DEPENDENT_TYPE` = 3

*Dependent variables.*

- INTEGER(INTG), parameter **EQUATIONS\_SET\_SETUP\_MATERIALS\_TYPE** = 4  
*Materials setup.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_SETUP\_SOURCE\_TYPE** = 5  
*Source setup.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_SETUP\_SOURCE\_MATERIALS\_TYPE** = 6  
*Source materials setup.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_SETUP\_ANALYTIC\_TYPE** = 7  
*Analytic setup.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_SETUP\_FIXED\_CONDITIONS\_TYPE** = 8  
*Fixed conditions.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_SETUP\_EQUATIONS\_TYPE** = 9  
*Equations setup.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_SETUP\_FINAL\_TYPE** = 9  
*Final setup.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_SETUP\_START\_ACTION** = 1  
*Start setup action.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_SETUP\_FINISH\_ACTION** = 2  
*Finish setup action.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_NOT\_FIXED** = 0  
*The dof is not fixed.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_FIXED\_BOUNDARY\_CONDITION** = 1  
*The dof is fixed as a boundary condition.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_MIXED\_BOUNDARY\_CONDITION** = 2  
*The dof is set as a mixed boundary condition.*
- INTEGER(INTG), parameter **NUMBER\_OF\_EQUATIONS\_SET\_LINEARITY\_TYPES** = 3  
*The number of problem linearity types defined.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_LINEAR** = 1  
*The problem is linear.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_NONLINEAR** = 2  
*The problem is non-linear.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_NONLINEAR\_BCS** = 3  
*The problem has non-linear boundary conditions.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_JACOBIAN\_NOT\_CALCULATED** = 1  
*The Jacobian values will not be calculated for the nonlinear equations set.*

- INTEGER(INTG), parameter **EQUATIONS\_SET\_JACOBIAN\_ANALYTIC\_CALCULATED** = 2  
*The Jacobian values will be calculated analytically for the nonlinear equations set.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_JACOBIAN\_FD\_CALCULATED** = 3  
*The Jacobian values will be calculated using finite differences for the nonlinear equations set.*
- INTEGER(INTG), parameter **NUMBER\_OF\_EQUATIONS\_SET\_TIME\_TYPES** = 3  
*The number of problem time dependence types defined.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_STATIC** = 1  
*The problem is static and has no time dependence.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_DYNAMIC** = 2  
*The problem is dynamic.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_QUASISTATIC** = 3  
*The problem is quasi-static.*
- INTEGER(INTG), parameter **NUMBER\_OF\_EQUATIONS\_SET SOLUTION\_METHODS** = 6  
*The number of solution methods defined.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_FEM SOLUTION\_METHOD** = 1  
*Finite Element Method solution method.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_BEM SOLUTION\_METHOD** = 2  
*Boundary Element Method solution method.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_FD SOLUTION\_METHOD** = 3  
*Finite Difference solution method.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_FV SOLUTION\_METHOD** = 4  
*Finite Volume solution method.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_GFEM SOLUTION\_METHOD** = 5  
*Grid-based Finite Element Method solution method.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_GFV SOLUTION\_METHOD** = 6  
*Grid-based Finite Volume solution method.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_NO\_OUTPUT** = 0  
*No output.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_TIMING\_OUTPUT** = 1  
*Timing information output.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_MATRIX\_OUTPUT** = 2  
*All below and equation matrices output.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_ELEMENT\_MATRIX\_OUTPUT** = 3

*All below and Element matrices output.*

- INTEGER(INTG), parameter EQUATIONS\_SET\_SPARSE\_MATRICES = 1

*Use sparse matrices for the equations set.*

- INTEGER(INTG), parameter EQUATIONS\_SET\_FULL\_MATRICES = 2

*Use fully populated matrices for the equations set.*

### 6.17.1 Detailed Description

This module defines all constants shared across equations set routines.

### 6.17.2 Variable Documentation

#### 6.17.2.1 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_- ADVECTION\_DIFFUSION\_EQUATION\_TYPE = 6

Definition at line 80 of file equations\_set\_constants.f90.

Referenced by CLASSICAL\_FIELD\_ROUTINES::CL(), CLASSICAL\_FIELD\_- ROUTINES::CLASSICAL\_FIELD\_EQUATIONS\_SET\_SETUP(), CLASSICAL\_FIELD\_- ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_CALCULATE(), CLASSICAL\_- FIELD\_ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_JACOBIAN\_EVALUATE(), and CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_RESIDUAL\_- EVALUATE().

#### 6.17.2.2 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_- BIHARMONIC\_EQUATION\_TYPE = 8

Definition at line 82 of file equations\_set\_constants.f90.

Referenced by CLASSICAL\_FIELD\_ROUTINES::CL(), CLASSICAL\_FIELD\_- ROUTINES::CLASSICAL\_FIELD\_EQUATIONS\_SET\_SETUP(), CLASSICAL\_FIELD\_- ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_CALCULATE(), CLASSICAL\_- FIELD\_ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_JACOBIAN\_EVALUATE(), and CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_RESIDUAL\_- EVALUATE().

#### 6.17.2.3 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_- CLASSICAL\_FIELD\_CLASS = 4

Definition at line 56 of file equations\_set\_constants.f90.

Referenced by CLASSICAL\_FIELD\_ROUTINES::CL(), LAPLACE\_EQUATIONS\_- ROUTINES::LAPLACE\_EQUATION\_EQUATIONS\_SET\_SUBTYPE\_SET(), and LAPLACE\_- EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_PROBLEM\_STANDARD\_SETUP().

---

Generated on Tue Nov 4 15:49:01 2008 for openCMISS by Doxygen

**6.17.2.4 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
CONSTANT\_SOURCE\_POISSON\_SUBTYPE = 1**

Definition at line 98 of file equations\_set\_constants.f90.

**6.17.2.5 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
DIFFUSION\_EQUATION\_TYPE = 5**

Definition at line 79 of file equations\_set\_constants.f90.

Referenced by CLASSICAL\_FIELD\_ROUTINES::CL(), CLASSICAL\_FIELD\_-ROUTINES::CLASSICAL\_FIELD\_EQUATIONS\_SET\_SETUP(), CLASSICAL\_FIELD\_-ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_CALCULATE(), CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_JACOBIAN\_EVALUATE(), and CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_RESIDUAL\_-EVALUATE().

**6.17.2.6 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
ELASTICITY\_CLASS = 1**

Definition at line 53 of file equations\_set\_constants.f90.

Referenced by FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITY\_EQUATIONS\_SET\_-SUBTYPE\_SET(), FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITY\_PROBLEM\_SETUP(), LINEAR\_ELASTICITY\_ROUTINES::LINEAR\_ELASTICITY\_EQUATIONS\_SET\_SUBTYPE\_SET(), and LINEAR\_ELASTICITY\_ROUTINES::LINEAR\_ELASTICITY\_PROBLEM\_SETUP().

**6.17.2.7 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
ELECTROMAGNETICS\_CLASS = 3**

Definition at line 55 of file equations\_set\_constants.f90.

**6.17.2.8 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
ELECTROSTATIC\_TYPE = 1**

Definition at line 71 of file equations\_set\_constants.f90.

**6.17.2.9 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
FINITE\_ELASTICITY\_TYPE = 2**

Definition at line 66 of file equations\_set\_constants.f90.

Referenced by ELASTICITY\_ROUTINES::EL(), ELASTICITY\_ROUTINES::ELASTICITY\_-EQUATIONS\_SET\_SETUP(), ELASTICITY\_ROUTINES::ELASTICITYFINITE\_ELEMENT\_-CALCULATE(), ELASTICITY\_ROUTINES::ELASTICITYFINITE\_ELEMENT\_JACOBIAN\_-EVALUATE(), ELASTICITY\_ROUTINES::ELASTICITYFINITE\_ELEMENT\_RESIDUAL\_-EVALUATE(), ELASTICITY\_ROUTINES::ELASTICITY\_PROBLEM\_CLASS\_TYPE\_SET(), ELASTICITY\_ROUTINES::ELASTICITY\_PROBLEM\_SETUP(), FINITE\_ELASTICITY\_-ROUTINES::FINITE\_ELASTICITY\_EQUATIONS\_SET\_SUBTYPE\_SET(), and FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITY\_PROBLEM\_SETUP().

---

**6.17.2.10 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
FITTING\_CLASS = 6**

Definition at line 59 of file equations\_set\_constants.f90.

**6.17.2.11 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
FLUID\_MECHANICS\_CLASS = 2**

Definition at line 54 of file equations\_set\_constants.f90.

**6.17.2.12 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
GENERALISED\_LAPLACE\_SUBTYPE = 2**

Definition at line 96 of file equations\_set\_constants.f90.

Referenced by LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_EQUATIONS\_SET\_-SETUP(), LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_EQUATIONS\_SET\_-SUBTYPE\_SET(), and LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATIONFINITE\_ELEMENT\_CALCULATE().

**6.17.2.13 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
HELMHOLTZ\_EQUATION\_TYPE = 3**

Definition at line 77 of file equations\_set\_constants.f90.

Referenced by CLASSICAL\_FIELD\_ROUTINES::CL(), CLASSICAL\_FIELD\_-ROUTINES::CLASSICAL\_FIELD\_EQUATIONS\_SET\_SETUP(), CLASSICAL\_FIELD\_-ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_CALCULATE(), CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_JACOBIAN\_EVALUATE(), and CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_RESIDUAL\_-EVALUATE().

**6.17.2.14 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
LAPLACE\_EQUATION\_TYPE = 1**

Definition at line 75 of file equations\_set\_constants.f90.

Referenced by CLASSICAL\_FIELD\_ROUTINES::CL(), CLASSICAL\_FIELD\_-ROUTINES::CLASSICAL\_FIELD\_EQUATIONS\_SET\_SETUP(), CLASSICAL\_FIELD\_-ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_CALCULATE(), CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_JACOBIAN\_EVALUATE(), CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_RESIDUAL\_EVALUATE(), LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_EQUATIONS\_SET\_SUBTYPE\_SET(), and LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_PROBLEM\_STANDARD\_SETUP().

**6.17.2.15 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
LINEAR\_ELASTIC\_MODAL\_TYPE = 1**

Definition at line 84 of file equations\_set\_constants.f90.

**6.17.2.16 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
LINEAR\_ELASTICITY\_TYPE = 1**

Definition at line 65 of file equations\_set\_constants.f90.

Referenced by ELASTICITY\_ROUTINES::EL(), ELASTICITY\_ROUTINES::ELASTICITY\_-  
EQUATIONS\_SET\_SETUP(), ELASTICITY\_ROUTINES::ELASTICITYFINITE\_ELEMENT\_-  
CALCULATE(), ELASTICITY\_ROUTINES::ELASTICITYFINITE\_ELEMENT\_JACOBIAN\_-  
EVALUATE(), ELASTICITY\_ROUTINES::ELASTICITYFINITE\_ELEMENT\_RESIDUAL\_-  
EVALUATE(), ELASTICITY\_ROUTINES::ELASTICITY\_PROBLEM\_CLASS\_TYPE\_SET(),  
ELASTICITY\_ROUTINES::ELASTICITY\_PROBLEM\_SETUP(), LINEAR\_ELASTICITY\_-  
ROUTINES::LINEAR\_ELASTICITY\_EQUATIONS\_SET\_SUBTYPE\_SET(), and LINEAR\_-  
ELASTICITY\_ROUTINES::LINEAR\_ELASTICITY\_PROBLEM\_SETUP().

**6.17.2.17 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
LINEAR\_SOURCE\_POISSON\_SUBTYPE = 2**

Definition at line 99 of file equations\_set\_constants.f90.

**6.17.2.18 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
MAGNETOSTATIC\_TYPE = 2**

Definition at line 72 of file equations\_set\_constants.f90.

**6.17.2.19 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
MAXWELLS\_EQUATIONS\_TYPE = 3**

Definition at line 73 of file equations\_set\_constants.f90.

**6.17.2.20 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
MODAL\_CLASS = 5**

Definition at line 58 of file equations\_set\_constants.f90.

**6.17.2.21 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
NAVIER\_STOKES\_FLUID\_TYPE = 2**

Definition at line 69 of file equations\_set\_constants.f90.

**6.17.2.22 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
NO\_CLASS = 0**

Definition at line 51 of file equations\_set\_constants.f90.

**6.17.2.23 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
NO\_SUBTYPE = 0**

Definition at line 87 of file equations\_set\_constants.f90.

Referenced by FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITY\_EQUATIONS\_SET\_SETUP(), FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITY\_EQUATIONS\_SET\_SUBTYPE\_Set(), FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITY\_PROBLEM\_SETUP(), LINEAR\_ELASTICITY\_ROUTINES::LINEAR\_ELASTICITY\_EQUATIONS\_SET\_SETUP(), LINEAR\_ELASTICITY\_ROUTINES::LINEAR\_ELASTICITY\_EQUATIONS\_SET\_SUBTYPE\_Set(), and LINEAR\_ELASTICITY\_ROUTINES::LINEAR\_ELASTICITY\_PROBLEM\_SETUP().

#### **6.17.2.24 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_NO\_TYPE = 0**

Definition at line 63 of file equations\_set\_constants.f90.

#### **6.17.2.25 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_OPTIMISATION\_CLASS = 7**

Definition at line 60 of file equations\_set\_constants.f90.

#### **6.17.2.26 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_POISSON\_EQUATION\_TYPE = 2**

Definition at line 76 of file equations\_set\_constants.f90.

Referenced by CLASSICAL\_FIELD\_ROUTINES::CL(), CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_EQUATIONS\_SET\_SETUP(), CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_CALCULATE(), CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_JACOBIAN\_EVALUATE(), and CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_RESIDUAL\_EVALUATE().

#### **6.17.2.27 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_QUADRATIC\_SOURCE\_POISSON\_SUBTYPE = 2**

Definition at line 100 of file equations\_set\_constants.f90.

#### **6.17.2.28 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SETREACTION\_DIFFUSION\_EQUATION\_TYPE = 7**

Definition at line 81 of file equations\_set\_constants.f90.

Referenced by CLASSICAL\_FIELD\_ROUTINES::CL(), CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_EQUATIONS\_SET\_SETUP(), CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_CALCULATE(), CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_JACOBIAN\_EVALUATE(), and CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_RESIDUAL\_EVALUATE().

#### **6.17.2.29 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SETSTANDARD\_LAPLACE\_SUBTYPE = 1**

Definition at line 95 of file equations\_set\_constants.f90.

Referenced by LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_EQUATIONS\_SET\_SETUP(), LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_EQUATIONS\_SET\_STANDARD\_SETUP(), LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_EQUATIONS\_SET\_SUBTYPE\_SET(), LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATIONFINITE\_ELEMENT\_CALCULATE(), and LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_PROBLEM\_STANDARD\_SETUP().

#### **6.17.2.30 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_STOKES\_FLUID\_TYPE = 1**

Definition at line 68 of file equations\_set\_constants.f90.

#### **6.17.2.31 INTEGER(INTG),parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_WAVE\_EQUATION\_TYPE = 4**

Definition at line 78 of file equations\_set\_constants.f90.

Referenced by CLASSICAL\_FIELD\_ROUTINES::CL(), CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_EQUATIONS\_SET\_SETUP(), CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_CALCULATE(), CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_JACOBIAN\_EVALUATE(), and CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELDFINITE\_ELEMENT\_RESIDUAL\_EVALUATE().

## 6.18 F90C Namespace Reference

This module handles calling C from Fortran90 and vise versa. It needs to be linked with the [f90c\\_c.c](#) module.

### Classes

- interface [interface](#)

### Functions

- INTEGER(INTG) [CSTRINGLENGTH](#) (CSTRING)
- subroutine [C2FSTRING](#) (CSTRING, FSTRING, ERR, ERROR,\*)
- INTEGER(INTG) [FSTRINGLENGTH](#) (FSTRING)
- subroutine [F2CSTRING](#) (CSTRING, FSTRING, ERR, ERROR,\*, FSTRINGLEN)

#### 6.18.1 Detailed Description

This module handles calling C from Fortran90 and vise versa. It needs to be linked with the [f90c\\_c.c](#) module.

#### 6.18.2 Function Documentation

##### 6.18.2.1 subroutine [F90C::C2FSTRING](#) (INTEGER(INTG),dimension(\*),intent(in) *CSTRING*, CHARACTER(LEN=\*),intent(out) *FSTRING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Definition at line 110 of file f90c\_f.f90.

References [CSTRINGLENGTH\(\)](#), [BASE\\_ROUTINES::ENTERS\(\)](#), [BASE\\_ROUTINES::ERRORS\(\)](#), and [BASE\\_ROUTINES::EXITS\(\)](#).

Referenced by [BINARY\\_FILE::CLOSE\\_BINARY\\_FILE\(\)](#), [TIMER::CPU\\_TIMER\(\)](#), [BINARY\\_FILE::INQUIRE\\_EOF\\_BINARY\\_FILE\(\)](#), [BINARY\\_FILE::OPEN\\_BINARY\\_FILE\(\)](#), [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_CHARACTER\(\)](#), [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_DP\(\)](#), [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_DP1\(\)](#), [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_DPC\(\)](#), [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_DPC1\(\)](#), [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_INTG\(\)](#), [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_LINTG\(\)](#), [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_LINTG1\(\)](#), [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_LOGICAL\(\)](#), [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_LOGICAL1\(\)](#), [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_SINTG\(\)](#), [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_SINTG1\(\)](#), [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_SP\(\)](#), [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_SP1\(\)](#), [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_SPC\(\)](#), [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_SPC1\(\)](#), [BINARY\\_FILE::SET\\_BINARY\\_FILE\(\)](#), [BINARY\\_FILE::SKIP\\_BINARY\\_FILE\(\)](#), [BINARY\\_FILE::WRITE\\_BINARY\\_FILE\\_CHARACTER\(\)](#), [BINARY\\_FILE::WRITE\\_BINARY\\_FILE\\_DP\(\)](#), [BINARY\\_FILE::WRITE\\_BINARY\\_FILE\\_DP1\(\)](#), [BINARY\\_FILE::WRITE\\_BINARY\\_FILE\\_DPC\(\)](#), [BINARY\\_FILE::WRITE\\_BINARY\\_FILE\\_DPC1\(\)](#), [BINARY\\_FILE::WRITE\\_BINARY\\_FILE\\_INTG\(\)](#), [BINARY\\_FILE::WRITE\\_BINARY\\_FILE\\_LINTG\(\)](#), [BINARY\\_FILE::WRITE\\_BINARY\\_FILE\\_LINTG1\(\)](#), [BINARY\\_FILE::WRITE\\_BINARY\\_FILE\\_LOGICAL\(\)](#), [BINARY\\_FILE::WRITE\\_BINARY\\_FILE\\_LOGICAL1\(\)](#), [BINARY\\_FILE::WRITE\\_BINARY\\_FILE\\_SINTG\(\)](#), [BINARY\\_FILE::WRITE\\_BINARY\\_FILE\\_SINTG1\(\)](#),

BINARY\_FILE::WRITE\_BINARY\_FILE\_SP(),      BINARY\_FILE::WRITE\_BINARY\_FILE\_SP1(),  
BINARY\_FILE::WRITE\_BINARY\_FILE\_SPC(),      and    BINARY\_FILE::WRITE\_BINARY\_FILE\_-  
SPC1().

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.18.2.2 INTEGER(INTG) F90C::CSTRINGLENGTH (INTEGER(INTG),dimension(\*),intent(in) CSTRING)

Definition at line 85 of file f90c\_f.f90.

Referenced by C2FSTRING(), and BINARY\_FILE::READ\_BINARY\_FILE\_CHARACTER().

Here is the caller graph for this function:

#### 6.18.2.3 subroutine F90C::F2CString (INTEGER(INTG),dimension(:,intent(out) CSTRING, CHARACTER(LEN=\*),intent(in) FSTRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*, INTEGER(INTG),intent(in),optional FSTRINGLEN)

Definition at line 174 of file f90c\_f.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-  
ROUTINES::EXITS(), FSTRINGLENGTH(), and MACHINE\_CONSTANTS::INTEGER\_SIZE.

Referenced by BINARY\_FILE::OPEN\_BINARY\_FILE(), and BINARY\_FILE::WRITE\_BINARY\_-  
FILE\_CHARACTER().

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.18.2.4 INTEGER(INTG) F90C::FSTRINGLENGTH (CHARACTER(LEN=\*),intent(in) FSTRING)

Definition at line 151 of file f90c\_f.f90.

Referenced by F2CString().

Here is the caller graph for this function:

## 6.19 FIELD\_IO\_ROUTINES Namespace Reference

Implements lists of Field IO operation.

### Classes

- struct [MESH\\_ELEMENTS\\_TYPE\\_PTR\\_TYPE](#)  
*field variable component type pointer for IO*
- struct [FIELD\\_VARIABLE\\_COMPONENT\\_PTR\\_TYPE](#)  
*field variable component type pointer for IO*
- struct [FIELD\\_IO\\_INFO\\_SET](#)  
*contains information for parallel IO, and it is nodal base*
- struct [FIELD\\_IO\\_NODAL\\_INFO\\_SET](#)  
*contains information for parallel IO, and it is nodal base*
- struct [FIELD\\_IO\\_ELEMENTALL\\_INFO\\_SET](#)  
*contains information for parallel IO, and it is nodal base*

### Functions

- subroutine [FIELD\\_IO\\_FIELD\\_INFO](#) (STRING, LABEL\_TYPE, FIELD\_TYPE, ERR, ERROR,\*)  
*Get the field information.*
- INTEGER(INTG) [FIELD\\_IO\\_DERIVATIVE\\_INFO](#) (LINE, ERR, ERROR)  
*Get the derivative information.*
- subroutine [FIELD\\_IO\\_CREATE\\_FIELDS](#)  
*Create decompsition.*
- subroutine [FIE](#) (DECOMPOSITION, DECOMPOSITION\_USER\_NUMBER, DECOMPOSITION\_METHOD, MESH, NUMBER\_OF\_DOMAINS,&ERR, ERROR,\*)  
*Create decompiton.*
- subroutine [FIELD\\_IO\\_FILEDS\\_IMPORT](#) (NAME, METHOD, REGION, MESH, MESH\_USER\_NUMBER, DECOMPOSITION, DECOMPOSITION\_USER\_NUMBER,&DECOMPOSITION\_METHOD, FIELD\_VALUES\_SET\_TYPE, FIELD\_SCALING\_TYPE, ERR, ERROR,\*)  
*Import fields from files into different computational nodes.*
- subroutine [FIELD\\_IO\\_FILL\\_BASIS\\_INFO](#) (INTERPOLATION\_XI, LIST\_STR, NUMBER\_OF\_COMPONENTS, ERR, ERROR,\*)  
*Finding basis information.*

- subroutine `FIELD_IO_IMPORT_GLOBAL_MESH` (NAME, REGION, MESH, MESH\_USER\_NUMBER, MESH\_TER\_COMPUTATIONAL\_NUMBER,&my\_computational\_node\_number,&MESH\_COMPONENTS\_OF\_FIELD\_COMPONENTS,&COMPONENTS\_IN\_FIELDS, NUMBER\_OF\_FIELDS, NUMBER\_OF\_EXNODE\_FILES, ERR, ERROR,\*)

*Read the global mesh into one computational node first and then broadcasting to others nodes.*

- subroutine `FIELD_IO_TRANSLATE_LABEL_INTO_INTERPOLATION_TYPE` (INTERPOLATION, LABEL\_TYPE, ERR, ERROR,\*)

*Finding basis information.*

- subroutine `FIELD_IO_ELEMENTS_EXPORT` (FIELDS, FILE\_NAME, METHOD, ERR, ERROR,\*)

*Export elemental information into multiple files.*

- TYPE(`VARYING_STRING`) `FIELD_IO_BASIS_LHTP_FAMILY_LABEL` (BASIS, num\_scl, num\_node, LABEL\_TYPE, ERR, ERROR)

*Finding basis information.*

- subroutine `FIELD_IO_EXPORT_ELEMENTAL_GROUP_HEADER_FORTRAN` (LOCAL\_PROCESS\_ELEMENTAL\_INFO\_SET, LOCAL ELEMENTAL\_NUMBER, MAX\_NODE\_CPMP\_INDEX,&NUM\_OF\_SCALING\_FACTOR\_SETS, LIST\_COMP\_SCALE, my\_computational\_node\_number, FILE\_ID, ERR, ERROR,\*)

*Write the header of a group elements using FORTRAN.*

- subroutine `FIELD_IO_EXPORT_ELEMENTS_INTO_` (LOCAL\_PROCESS\_ELEMENTAL\_INFO\_SET, NAME, my\_computational\_node\_number,&computational\_node\_numbers, ERR, ERROR,\*)

*Write all the elemental information from LOCAL\_PROCESS\_NODAL\_INFO\_SET to exelem files.*

- subroutine `FIELD_IO_ELEMENTAL_INFO_SET_SORT` (LOCAL\_PROCESS\_ELEMENTAL\_INFO\_SET, my\_computational\_node\_number, ERR, ERROR,\*)

*Sort the Elemental\_info\_set according to the type of field variable components.*

- subroutine `FIELD_IO_ELEMENTAL_INFO_SET_ATTACH_LOCAL_PROCESS` (LOCAL\_PROCESS\_ELEMENTAL\_INFO\_SET, ERR, ERROR,\*)

*Collect the elemental information from each MPI process.*

- subroutine `FIELD_IO_ELEMENTAL_INFO_SET_FINALIZE` (LOCAL\_PROCESS\_ELEMENTAL\_INFO\_SET, ERR, ERROR,\*)

*Finalized the elemental information set.*

- subroutine `FIELD_IO_ELEMENTAL_INFO_SET_INITIALISE` (LOCAL\_PROCESS\_ELEMENTAL\_INFO\_SET, FIELDS, ERR, ERROR,\*)

*Initialize the elemental information set.*

- subroutine `FIELD_IO_NODES_EXPORT` (FIELDS, FILE\_NAME, METHOD, ERR, ERROR,\*)

*Export nodal information.*

- subroutine `FIELD_IO_NODAL_INFO_SET_INITIALISE` (LOCAL\_PROCESS\_NODAL\_INFO\_SET, FIELDS, ERR, ERROR,\*)

*Initialize nodal information set.*

- subroutine [FIELD\\_IO\\_NODAL\\_INFO\\_SET\\_ATTACH\\_LOCAL\\_PROCESS](#) (LOCAL\_PROCESS\_NODAL\_INFO\_SET, ERR, ERROR,\*)
 

*Collect nodal information from each MPI process.*
- subroutine [FIELD\\_IO\\_NODAL\\_INFO\\_SET\\_SORT](#) (LOCAL\_PROCESS\_NODAL\_INFO\_SET, my\_computational\_node\_number, ERR, ERROR,\*)
 

*Sort nodal information according to the type of field variable component.*
- subroutine [FIELD\\_IO\\_NODAL\\_INFO\\_SET\\_FINALIZE](#) (LOCAL\_PROCESS\_NODAL\_INFO\_SET, ERR, ERROR,\*)
 

*Finalize nodal information set.*
- TYPE(VARYING\_STRING) [FIELD\\_IO\\_LABEL\\_DERIVATIVE\\_INFO\\_GET](#) (GROUP\_DERIVATIVES, NUMBER\_DERIVATIVES, LABEL\_TYPE, ERR, ERROR)
 

*Get the derivative information.*
- TYPE(VARYING\_STRING) [FIELD\\_IO\\_LABEL\\_FIELD\\_INFO\\_GET](#) (COMPONENT, LABEL\_TYPE, ERR, ERROR)
 

*Get the field information.*
- subroutine [FIELD\\_IO\\_EXPORT\\_NODAL\\_GROUP\\_HEADER\\_FORTRA](#) (LOCAL\_PROCESS\_NODAL\_INFO\_SET, LOCAL\_NODAL\_NUMBER, MAX\_NUM\_OF\_NODAL\_DERIVATIVES,&my\_computational\_node\_number, FILE\_ID, ERR, ERROR,\*)
 

*Write the header of a group nodes using FORTRAN.*
- subroutine [FIELD\\_IO\\_EXPORT\\_NODES\\_INTO\\_LOC](#) (LOCAL\_PROCESS\_NODAL\_INFO\_SET, NAME, my\_computational\_node\_number,&computational\_node\_numbers, ERR, ERROR,\*)
 

*Write all the nodal information from LOCAL\_PROCESS\_NODAL\_INFO\_SET to local exnode files.*
- TYPE(VARYING\_STRING) [FIELD\\_IO\\_FILEDS\\_GROUP\\_INFO\\_GET](#) (FIELDS, ERR, ERROR)
 

*Get the region label.*
- TYPE(VARYING\_STRING) [FIELD\\_IO\\_MULTI\\_FILES\\_INFO\\_GET](#) (computational\_node\_numbers, ERR, ERROR)
 

*Get the number of files.*
- subroutine [FIELD\\_IO\\_FORTRAN\\_FILE\\_READ\\_STRING](#) (FILE\_ID, STRING\_DATA, FILE-END, ERR, ERROR,\*)
 

*Read a string using FORTRAN IO.*
- subroutine [FIELD\\_IO\\_FORTRAN\\_FILE\\_WRITE\\_STRING](#) (FILE\_ID, STRING\_DATA, LEN\_OF\_DATA, ERR, ERROR,\*)
 

*Write a string using FORTRAN IO.*
- subroutine [FIELD\\_IO\\_FORTRAN\\_FILE\\_READ\\_DP](#) (FILE\_ID, REAL\_DATA, LEN\_OF\_DATA, FILE-END, ERR, ERROR,\*)
 

*Read a real data using FORTRAN IO.*

- subroutine `FIELD_IO_FORTRAN_FILE_WRITE_DP` (FILE\_ID, REAL\_DATA, LEN\_OF\_DATA, ERR, ERROR,\*)
 

*Write a real data using FORTRAN IO.*
- subroutine `FIELD_IO_FORTRAN_FILE_READ_INTG` (FILE\_ID, INTG\_DATA, LEN\_OF\_DATA, ERR, ERROR,\*)
 

*Read a integer data.*
- subroutine `FIELD_IO_FORTRAN_FILE_WRITE_INTG` (FILE\_ID, INTG\_DATA, LEN\_OF\_DATA, ERR, ERROR,\*)
 

*Write a integer data.*
- subroutine `FIELD_IO_FORTRAN_FILE_OPEN` (FILE\_ID, FILE\_NAME, FILE\_STATUS, ERR, ERROR,\*)
 

*Open a file using Fortran.*
- subroutine `FIELD_IO_FORTRAN_FILE_REWIND` (FILE\_ID, ERR, ERROR,\*)
 

*Close a file using FORTRAN IO to rewind.*
- subroutine `FIELD_IO_FORTRAN_FILE_CLOSE` (FILE\_ID, ERR, ERROR,\*)
 

*Close a file using Fortran.*
- subroutine `STRING_TO_MUTI_INTEGERS_VS` (STRING, NUMBER\_OF\_INTEGERS, INTG\_DATA, ERR, ERROR,\*)
- subroutine `STRING_TO_MUTI_REALS_VS` (STRING, NUMBER\_OF\_REALS, REAL\_DATA, POSITION, ERR, ERROR,\*)

## Variables

- INTEGER(INTG), parameter `SHAPE_SIZE` = 3
 

*size of shape*
- INTEGER(INTG), parameter `FIELD_IO_FIELD_LABEL` = 1
 

*Type for label.*
- INTEGER(INTG), parameter `FIELD_IO_VARIABLE_LABEL` = 2
- INTEGER(INTG), parameter `FIELD_IO_COMPONENT_LABEL` = 3
- INTEGER(INTG), parameter `FIELD_IO_DERIVATIVE_LABEL` = 4
- INTEGER(INTG), parameter `FIELD_IO_SCALE_FACTORS_NUMBER_TYPE` = 5
 

*Type of scale factor.*
- INTEGER(INTG), parameter `FIELD_IO_SCALE_FACTORS_PROPERTY_TYPE` = 6

### 6.19.1 Detailed Description

Implements lists of Field IO operation.

## 6.19.2 Function Documentation

**6.19.2.1 subroutine FIELD\_IO\_ROUTINES::FIE** (TYPE(DECOMPOSITION\_TYPE),pointer *DECOMPOSITION*, INTEGER(INTG),intent(in) *DECOMPOSITION\_USER\_NUMBER*, INTEGER(INTG),intent(in) *DECOMPOSITION\_METHOD*, TYPE(MESH\_-TYPE),pointer *MESH*, INTEGER(INTG),intent(in) *NUMBER\_OF\_DOMAINS*, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, \*) [private]

Create decomposition.

### Parameters:

*DECOMPOSITION* decomposition type

*DECOMPOSITION\_USER\_NUMBER* user number for decomposition

*DECOMPOSITION\_METHOD* decomposition type

*MESH* mesh type

*NUMBER\_OF\_DOMAINS* number of domains

Definition at line 690 of file field\_IO\_routines.f90.

References MESH\_ROUTINES::DECOMPOSITION\_CREATE\_FINISH(), MESH\_-ROUTINES::DECOMPOSITION\_CREATE\_START(), MESH\_ROUTINES::DECOMPOSITION\_-NUMBER\_OF\_DOMAINS\_SET(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.19.2.2 TYPE(VARYING\_STRING) FIELD\_IO\_ROUTINES::FIELD\_IO\_-BASIS\_LHTP\_FAMILY\_LABEL** (TYPE(BASIS\_TYPE),intent(in) *BASIS*, INTEGER(INTG),intent(inout) *num\_scl*, INTEGER(INTG),intent(inout) *num\_node*, INTEGER(INTG),intent(in) *LABEL\_TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]

Finding basis information.

### Parameters:

*BASIS* The error string

*ERR* The error code

*ERROR* The error string

Definition at line 1853 of file field\_IO\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.19.2.3 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_CREATE\_FIELDS()** [private]

Create decomposition.

Definition at line 252 of file field\_IO\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `FIELD_ROUTINES::FIELD_CREATE_FINISH()`,    `FIELD_ROUTINES::FIELD_VALUES_SET_TYPE()`, `CMISS_MPI::MPI_ERROR_CHECK()`, and `KINDS::PTR`.

Here is the call graph for this function:

**6.19.2.4 INTEGER(INTG) FIELD\_IO\_ROUTINES::FIELD\_IO\_DERIVATIVE\_INFO  
 (TYPE(VARYING\_STRING),intent(in) *LINE*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]**

Get the derivative information.

**Parameters:**

***LINE*** Text info  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 187 of file `field_IO_routines.f90`.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.19.2.5 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_ELEMENTAL\_INFO\_SET\_ATTACH\_LOCAL\_PROCESS (TYPE(FIELD\_IO\_ELEMENTALL\_INFO\_SET),intent(inout)  
 LOCAL\_PROCESS\_ELEMENTAL\_INFO\_SET, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Collect the elemental information from each MPI process.

**Parameters:**

***LOCAL\_PROCESS\_ELEMENTAL\_INFO\_SET*** nodal information in this process  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 2765 of file `field_IO_routines.f90`.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `KINDS::PTR`.

Here is the call graph for this function:

**6.19.2.6 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_ELEMENTAL\_INFO\_SET\_FINALIZE (TYPE(FIELD\_IO\_ELEMENTALL\_INFO\_SET),intent(inout)  
 LOCAL\_PROCESS\_ELEMENTAL\_INFO\_SET, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Finalized the elemental information set.

**Parameters:**

***LOCAL\_PROCESS\_ELEMENTAL\_INFO\_SET*** nodal information in this process

**ERR** The error code

**ERROR** The error string

Definition at line 2951 of file field\_IO\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `KINDS::PTR`.

Here is the call graph for this function:

```
6.19.2.7 subroutine FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_-
 INITIALISE (TYPE(FIELD_IO_ELEMENTALL_INFO_SET),intent(inout)
 LOCAL_PROCESS_ELEMENTAL_INFO_SET, TYPE(FIELDS_TYPE),pointer
 FIELDS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)
 ERROR, *) [private]
```

Initialize the elemental information set.

#### Parameters:

`LOCAL_PROCESS_ELEMENTAL_INFO_SET` elemental information in this process

`FIELDS` the field object

**ERR** The error code

**ERROR** The error string

Definition at line 2996 of file field\_IO\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `KINDS::PTR`.

Here is the call graph for this function:

```
6.19.2.8 subroutine FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_-
 SET_SORT (TYPE(FIELD_IO_ELEMENTALL_INFO_SET),intent(inout)
 LOCAL_PROCESS_ELEMENTAL_INFO_SET, INTEGER(INTG),intent(in)
 my_computational_node_number, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Sort the Elemental\_info\_set according to the type of field variable components.

#### Parameters:

`LOCAL_PROCESS_ELEMENTAL_INFO_SET` elemental information in this process

`my_computational_node_number` local process number

**ERR** The error code

**ERROR** The error string

Definition at line 2534 of file field\_IO\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `KINDS::PTR`.

Here is the call graph for this function:

---

**6.19.2.9 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_ELEMENTS\_EXPORT**  
 (TYPE(FIELDS\_TYPE),pointer *FIELDS*, TYPE(VARYING\_STRING),intent(inout)  
*FILE\_NAME*, TYPE(VARYING\_STRING),intent(in) *METHOD*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Export elemental information into multiple files.

**See also:**

{FIELD\_IO::FIELD\_IO\_ELEMENTS\_EXPORT}.

**Parameters:**

*FIELDS* the field object  
*FILE\_NAME* file name  
*METHOD* method used for IO  
*ERR* The error code  
*ERROR* The error string

Definition at line 1802 of file field\_IO\_routines.f90.

References COMP\_ENVIRONMENT::COMPUTATIONAL\_NODE\_NUMBER\_GET(), COMP\_ENVIRONMENT::COMPUTATIONAL\_NODES\_NUMBER\_GET(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.19.2.10 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_EXPORT\_ELEMENTAL\_GROUP\_-  
 HEADER\_FORTRAN**(TYPE(FIELD\_IO\_ELEMENTALL\_INFO\_SET),intent(inout)  
*LOCAL\_PROCESS\_ELEMENTAL\_INFO\_SET*, LOCAL ELEMENTAL\_NUMBER,  
 INTEGER(INTG),intent(inout) *MAX\_NODE\_CPMP\_INDEX*, &,intent(inout)  
*NUM\_OF\_SCALING\_FACTOR\_SETS*, INTEGER(INTG),dimension(:),intent(inout)  
*LIST\_COMP\_SCALE*, INTEGER(INTG),intent(in) *my\_computational\_node\_number*,  
 INTEGER(INTG),intent(in) *FILE\_ID*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Write the header of a group elements using FORTRAN.

**Parameters:**

*LOCAL\_PROCESS\_ELEMENTAL\_INFO\_SET* LOCAL\_PROCESS\_NODAL\_INFO\_SET  
*MAX\_NODE\_CPMP\_INDEX* MAX\_NODE\_INDEX  
*my\_computational\_node\_number* local process number  
*FILE\_ID* FILE ID  
*ERR* The error code  
*ERROR* The error string

Definition at line 1970 of file field\_IO\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

---

**6.19.2.11 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_EXPORT\_ELEMENTS\_INTO\_(  
 (TYPE(FIELD\_IO\_ELEMENTALL\_INFO\_SET),intent(inout)  
 LOCAL\_PROCESS\_ELEMENTAL\_INFO\_SET, TYPE(VARYING\_STRING),intent(in)  
 NAME, INTEGER(INTG),intent(in) my\_computational\_node\_number,  
 &,intent(in) computational\_node\_numbers, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Write all the elemental information from LOCAL\_PROCESS\_NODAL\_INFO\_SET to exelem files.

**Parameters:**

**LOCAL\_PROCESS\_ELEMENTAL\_INFO\_SET** nodal information in this process  
**NAME** the prefix name of file.  
**my\_computational\_node\_number** local process number  
**ERR** The error code  
**ERROR** The error string

Definition at line 2277 of file field\_IO\_routines.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_-ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

---

**6.19.2.12 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_EXPORT\_NODAL\_GROUP\_-  
 HEADER\_FORTRA (TYPE(FIELD\_IO\_NODAL\_INFO\_SET),intent(inout)  
 LOCAL\_PROCESS\_NODAL\_INFO\_SET, INTEGER(INTG),intent(in)  
 LOCAL\_NODAL\_NUMBER, INTEGER(INTG),intent(inout) MAX\_NUM\_-  
 OF\_NODAL\_DERIVATIVES, &,intent(in) my\_computational\_node\_number,  
 INTEGER(INTG),intent(in) FILE\_ID, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Write the header of a group nodes using FORTRAIN.

**Parameters:**

**LOCAL\_PROCESS\_NODAL\_INFO\_SET** LOCAL\_PROCESS\_NODAL\_INFO\_SET  
**LOCAL\_NODAL\_NUMBER** LOCAL\_NUMBER IN THE NODAL IO LIST  
**MAX\_NUM\_OF\_NODAL\_DERIVATIVES** MAX\_NUM\_OF\_NODAL\_DERIVATIVES  
**FILE\_ID** FILE ID  
**ERR** The error code  
**ERROR** The error string

Definition at line 4313 of file field\_IO\_routines.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_-ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

---

**6.19.2.13 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_EXPORT\_NODES\_-  
 INTO\_LOC (TYPE(FIELD\_IO\_NODAL\_INFO\_SET),intent(inout)  
 LOCAL\_PROCESS\_NODAL\_INFO\_SET, TYPE(VARYING\_STRING),intent(in)  
 NAME, INTEGER(INTG),intent(in) my\_computational\_node\_number,  
 &,intent(in) computational\_node\_numbers, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Write all the nodal information from LOCAL\_PROCESS\_NODAL\_INFO\_SET to local exnode files.

**Parameters:**

*LOCAL\_PROCESS\_NODAL\_INFO\_SET* nodal information in this process  
*NAME* the prefix name of file.  
*my\_computational\_node\_number* local process number  
*ERR* The error code  
*ERROR* The error string

Definition at line 5085 of file field\_IO\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), FIELD\_ROUTINES::FIELD\_VALUES\_SET\_TYPE, and KINDS::PTR.

Here is the call graph for this function:

**6.19.2.14 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_FIELD\_INFO (TYPE(VARYING\_-  
 STRING),intent(in) STRING, INTEGER(INTG),intent(in) LABEL\_TYPE,  
 INTEGER(INTG),intent(inout) FIELD\_TYPE, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Get the field information.

**Parameters:**

*LABEL\_TYPE* identitor for information  
*ERR* The error code  
*ERROR* The error string

Definition at line 138 of file field\_IO\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), FIELD\_ROUTINES::FIELD\_FIBRE\_TYPE, and FIELD\_ROUTINES::FIELD\_-GEOMETRIC\_TYPE.

Here is the call graph for this function:

**6.19.2.15 TYPE(VARYING\_STRING) FIELD\_IO\_ROUTINES::FIELD\_IO\_FILEDS\_GROUP\_-  
 INFO\_GET (TYPE(FIELDS\_TYPE),intent(in) FIELDS, INTEGER(INTG),intent(out)  
 ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

Get the region label.

**Parameters:**

*FIELDS* FILEDS

**ERR** The error code

**ERROR** The error string

Definition at line 5250 of file field\_IO\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.19.2.16 subroutine FIELD_IO_ROUTINES::FIELD_IO_FILEDS_IMPORT
 (TYPE(VARYING_STRING),intent(in) NAME, TYPE(VARYING_STRING),intent(in)
 METHOD, TYPE(REGION_TYPE),pointer REGION, TYPE(MESH_TYPE),pointer
 MESH, INTEGER(INTG),intent(in) MESH_USER_NUMBER,
 TYPE(DECOMPOSITION_TYPE),pointer DECOMPOSITION,
 INTEGER(INTG),intent(in) DECOMPOSITION_USER_NUMBER,
 &,intent(in) DECOMPOSITION_METHOD, INTEGER(INTG),intent(in)
 FIELD_VALUES_SET_TYPE, INTEGER(INTG),intent(in) FIELD_SCALING_TYPE,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *)
```

Import fields from files into different computational nodes.

#### Parameters:

**NAME** name of input

**METHOD** method used for import

**REGION** region

**MESH** mesh type

**MESH\_USER\_NUMBER** user number for mesh

**DECOMPOSITION** decompistion

**DECOMPOSITION\_USER\_NUMBER** user number for decompistion

**ERR** The error code

**ERROR** The error string

Definition at line 727 of file field\_IO\_routines.f90.

References `COMP_ENVIRONMENT::COMPUTATIONAL_NODE_NUMBER_GET()`, `COMP_ENVIRONMENT::COMPUTATIONAL_NODES_NUMBER_GET()`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `FIELD_ROUTINES::FIELD_VALUES_SET_TYPE`.

Here is the call graph for this function:

```
6.19.2.17 subroutine FIELD_IO_ROUTINES::FIELD_IO_FILL_BASIS_INFO
 (INTEGER(INTG),dimension(:, :, intent(inout) INTERPOLATION_XI,
 TYPE(VARYING_STRING),dimension(:, intent(inout) LIST_STR,
 INTEGER(INTG),intent(in) NUMBER_OF_COMPONENTS,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *) [private]
```

Finding basis information.

**Parameters:**

***INTERPOLATION\_XI*** xi interpolation type

***LIST\_STR*** label type

***ERR*** The error code

***ERROR*** The error string

Definition at line 798 of file field\_IO\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    and    **BASE\_ROUTINES::EXITS()**.

Here is the call graph for this function:

**6.19.2.18 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_FORTRAN\_FILE\_CLOSE  
 (INTEGER(INTG),intent(inout) *FILE\_ID*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Close a file using Fortran.

**Parameters:**

***FILE\_ID*** file ID

***ERR*** The error code

***ERROR*** The error string

Definition at line 5686 of file field\_IO\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    and    **BASE\_ROUTINES::EXITS()**.

Here is the call graph for this function:

**6.19.2.19 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_FORTRAN\_FILE\_OPEN  
 (INTEGER(INTG),intent(inout) *FILE\_ID*, TYPE(VARYING\_STRING),intent(inout)  
*FILE\_NAME*, TYPE(VARYING\_STRING),intent(in) *FILE\_STATUS*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]**

Open a file using Fortran.

**Parameters:**

***FILE\_ID*** file ID

***FILE\_NAME*** the name of file.

***FILE\_STATUS*** status for opening a file

***ERR*** The error code

***ERROR*** The error string

Definition at line 5603 of file field\_IO\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    and    **BASE\_ROUTINES::EXITS()**.

Here is the call graph for this function:

---

**6.19.2.20 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_FORTRAN\_FILE\_READ\_DP**  
 (INTEGER(INTG),intent(in) *FILE\_ID*, REAL(DP),dimension(:),intent(out)  
*REAL\_DATA*, INTEGER(INTG),intent(in) *LEN\_OF\_DATA*,  
 LOGICAL,intent(inout) *FILE\_END*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Read a real data using FORTRAN IO.

**Parameters:**

***FILE\_ID*** file ID  
***REAL\_DATA*** the name of file.  
***LEN\_OF\_DATA*** length of string  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 5446 of file field\_IO\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.19.2.21 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_FORTRAN\_FILE\_READ\_INTG**  
 (INTEGER(INTG),intent(in) *FILE\_ID*, INTEGER(INTG),dimension(:),intent(out)  
*INTG\_DATA*, INTEGER(INTG),intent(in) *LEN\_OF\_DATA*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Read a integer data.

**Parameters:**

***FILE\_ID*** file ID  
***INTG\_DATA*** the name of file.  
***LEN\_OF\_DATA*** length of string  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 5547 of file field\_IO\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.19.2.22 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_FORTRAN\_FILE\_READ\_STRING**  
 (INTEGER(INTG),intent(in) *FILE\_ID*, TYPE(VARYING\_STRING),intent(inout)  
*STRING\_DATA*, LOGICAL,intent(inout) *FILE\_END*, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Read a string using FORTRAN IO.

**Parameters:**

**FILE\_ID** file ID  
**STRING\_DATA** the string data.  
**FILE\_END** file end  
**ERR** The error code  
**ERROR** The error string

Definition at line 5335 of file field\_IO\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.19.2.23 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_FORTRAN\_FILE\_REWIND**  
`(INTEGER(INTG),intent(inout) FILE_ID, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Close a file using FORTRAN IO to rewind.

**Parameters:**

**FILE\_ID** file ID  
**ERR** The error code  
**ERROR** The error string

Definition at line 5662 of file field\_IO\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.19.2.24 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_FORTRAN\_FILE\_WRITE\_DP**  
`(INTEGER(INTG),intent(in) FILE_ID, REAL(DP),dimension(:),intent(in) REAL_-`  
`DATA, INTEGER(INTG),intent(in) LEN_OF_DATA, INTEGER(INTG),intent(out)`  
`ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Write a real data using FORTRAN IO.

**Parameters:**

**FILE\_ID** file ID  
**REAL\_DATA** the name of file.  
**LEN\_OF\_DATA** length of string  
**ERR** The error code  
**ERROR** The error string

Definition at line 5482 of file field\_IO\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

---

**6.19.2.25 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_FORTRAN\_FILE\_WRITE\_INTG**  
 (INTEGER(INTG),intent(in) *FILE\_ID*, INTEGER(INTG),dimension(:),intent(in)  
*INTG\_DATA*, INTEGER(INTG),intent(in) *LEN\_OF\_DATA*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Write a integer data.

**Parameters:**

***FILE\_ID*** file ID

***INTG\_DATA*** the name of file.

***LEN\_OF\_DATA*** length of string

***ERR*** The error code

***ERROR*** The error string

Definition at line 5575 of file field\_IO\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

---

**6.19.2.26 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_FORTRAN\_FILE\_WRITE\_STRING**  
 (INTEGER(INTG),intent(in) *FILE\_ID*, TYPE(VARYING\_STRING),intent(in)  
*STRING\_DATA*, INTEGER(INTG),intent(in) *LEN\_OF\_DATA*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Write a string using FORTRAN IO.

**Parameters:**

***FILE\_ID*** file ID

***STRING\_DATA*** the string data.

***LEN\_OF\_DATA*** length of string

***ERR*** The error code

***ERROR*** The error string

Definition at line 5378 of file field\_IO\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

6.19.2.27 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_IMPORT\_GLOBAL\_MESH  
 (TYPE(VARYING\_STRING),intent(in) NAME, TYPE(REGION\_TYPE),pointer  
*REGION*, TYPE(MESH\_TYPE),pointer *MESH*, INTEGER(INTG),intent(in)  
*MESH\_USER\_NUMBER*, INTEGER(COMPUTATIONAL\_NUMBER, &,intent(in)  
*my\_computational\_node\_number*, &,dimension(:),intent(inout),allocatable *MESH\_-  
COMPONENTS\_OF\_FIELD\_COMPONENTS*, &,dimension(:),intent(inout),allocatable  
*COMPONENTS\_IN\_FIELDS*, INTEGER(INTG),intent(inout) *NUMBER\_OF\_FIELDS*,  
INTEGER(INTG),intent(inout) *NUMBER\_OF\_EXNODE\_FILES*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
\*) [private]

Read the global mesh into one computational node first and then broadcasting to others nodes.

## Parameters:

*NAME* the name of elment file

*REGION* region

*MESH* mesh type

**MESH\_USER\_NUMBER** user number of mesh

**ERR** The error code

**ERROR** The error string

Definition at line 842 of file field\_IO\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `MESH_ROUTINES::MESH_CREATE_FINISH()`, `MESH_ROUTINES::MESH_CREATE_START()`,    `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_FINISH()`,    `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_START()`, `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_BASIS_SET()`,    `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_NODES_SET()`,    `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_NUMBER_SET()`,    `CMISS_MPI::MPI_ERROR_CHECK()`, `NODE_ROUTINES::NODE_NUMBER_SET()`, `NODE_ROUTINES::NODES_CREATE_FINISH()`, `NODE_ROUTINES::NODES_CREATE_START()`, and `KINDS::PTR`.

Here is the call graph for this function:

**6.19.2.28** **TYPE(VARYING\_STRING) FIELD\_IO\_ROUTINES::FIELD\_IO\_LABEL\_-  
DERIVATIVE\_INFO\_GET (INTEGER(INTG),dimension(number\_-  
derivatives),intent(in) *GROUP\_DERIVATIVES*, INTEGER(INTG),intent(in)  
*NUMBER\_DERIVATIVES*, INTEGER(INTG),intent(in) *LABEL\_TYPE*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*)  
[private]**

Get the derivative information.

## Parameters:

**LABEL\_TYPE** identifier for information

**ERR** The error code

**ERROR** The error string

Definition at line 3761 of file field\_IO\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.19.2.29** `TYPE(VARYING_STRING) FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET (TYPE(FIELD_VARIABLE_COMPONENT_TYPE),pointer COMPONENT, INTEGER(INTG),intent(in) LABEL_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)`  
`[private]`

Get the field information.

**Parameters:**

*LABEL\_TYPE* identitor for information  
*ERR* The error code  
*ERROR* The error string

Definition at line 3845 of file `field_IO_routines.f90`.

References `COORDINATE_ROUTINES::COORDINATE_RECTANGULAR_CARTESIAN_TYPE`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_ROUTINES::FIELD_FIBRE_TYPE`, `FIELD_ROUTINES::FIELD_GENERAL_TYPE`, `FIELD_ROUTINES::FIELD_GEOMETRIC_TYPE`, `FIELD_ROUTINES::FIELD_MATERIAL_TYPE`, `FIELD_ROUTINES::FIELD_NORMAL_VARIABLE_TYPE`, `FIELD_ROUTINES::FIELD_STANDARD_VARIABLE_TYPE`, `FIELD_ROUTINES::FIELD_TIME_DERIV1_VARIABLE_TYPE`, and `FIELD_ROUTINES::FIELD_TIME_DERIV2_VARIABLE_TYPE`.

Here is the call graph for this function:

**6.19.2.30** `TYPE(VARYING_STRING) FIELD_IO_ROUTINES::FIELD_IO_MULTI_FILES_INFO_GET (INTEGER(INTG),intent(in) computational_node_numbers, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)`  
`[private]`

Get the number of files.

**Parameters:**

*ERR* The error code  
*ERROR* The error string

Definition at line 5277 of file `field_IO_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.19.2.31** `subroutine FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_ATTACH_LOCAL_PROCESS (TYPE(FIELD_IO_NODAL_INFO_SET),intent(inout) LOCAL_PROCESS_NODAL_INFO_SET, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`  
`[private]`

Collect nodal information from each MPI process.

**Parameters:**

**LOCAL\_PROCESS\_NODAL\_INFO\_SET** nodal information in this process  
**ERR** The error code  
**ERROR** The error string

Definition at line 3272 of file field\_IO\_routines.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_-ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

**6.19.2.32 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_NODAL\_INFO\_SET\_FINALIZE (TYPE(FIELD\_IO\_NODAL\_INFO\_SET),intent(inout) LOCAL\_PROCESS\_NODAL\_INFO\_SET, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Finalize nodal information set.

**Parameters:**

**LOCAL\_PROCESS\_NODAL\_INFO\_SET** nodal information in this process  
**ERR** The error code  
**ERROR** The error string

Definition at line 3716 of file field\_IO\_routines.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_-ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

**6.19.2.33 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_NODAL\_INFO\_SET\_INITIALISE (TYPE(FIELD\_IO\_NODAL\_INFO\_SET),intent(inout) LOCAL\_PROCESS\_NODAL\_INFO\_SET, TYPE(FIELDS\_TYPE),pointer FIELDS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Initialize nodal information set.

**Parameters:**

**LOCAL\_PROCESS\_NODAL\_INFO\_SET** nodal information in this process  
**FIELDS** the field object  
**ERR** The error code  
**ERROR** The error string

Definition at line 3186 of file field\_IO\_routines.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_-ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

---

**6.19.2.34 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_NODAL\_INFO\_SET\_SORT**  
 (**TYPE(FIELD\_IO\_NODAL\_INFO\_SET),intent(inout)** *LOCAL\_PROCESS\_NODAL\_INFO\_SET*, **INTEGER(INTG),intent(in)** *my\_computational\_node\_number*,  
**INTEGER(INTG),intent(out)** *ERR*, **TYPE(VARYING\_STRING),intent(out)** *ERROR*,  
\*) [private]

Sort nodal information according to the type of field variable component.

**Parameters:**

*LOCAL\_PROCESS\_NODAL\_INFO\_SET* nodal information in this process  
*my\_computational\_node\_number* local process number  
*ERR* The error code  
*ERROR* The error string

Definition at line 3463 of file field\_IO\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    **BASE\_ROUTINES::EXITS()**, and **KINDS::PTR**.

Here is the call graph for this function:

**6.19.2.35 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_NODES\_EXPORT**  
 (**TYPE(FIELDS\_TYPE),pointer** *FIELDS*, **TYPE(VARYING\_STRING),intent(inout)** *FILE\_NAME*, **TYPE(VARYING\_STRING),intent(in)** *METHOD*,  
**INTEGER(INTG),intent(out)** *ERR*, **TYPE(VARYING\_STRING),intent(out)** *ERROR*,  
\*)

Export nodal information.

**See also:**

{FIELD\_IO::FIELD\_IO\_NODES\_EXPORT}.

**Parameters:**

*FIELDS* the field object  
*FILE\_NAME* file name  
*ERR* The error code  
*ERROR* The error string

Definition at line 3135 of file field\_IO\_routines.f90.

References    **COMP\_ENVIRONMENT::COMPUTATIONAL\_NODE\_NUMBER\_GET()**,    **COMP\_ENVIRONMENT::COMPUTATIONAL\_NODES\_NUMBER\_GET()**,    **BASE\_ROUTINES::ENTERS()**,  
**BASE\_ROUTINES::ERRORS()**, and **BASE\_ROUTINES::EXITS()**.

Here is the call graph for this function:

**6.19.2.36 subroutine FIELD\_IO\_ROUTINES::FIELD\_IO\_TRANSLATE\_LABEL\_INTO\_INTERPOLATION\_TYPE** (**INTEGER(INTG),intent(inout)** *INTERPOLATION*,  
**TYPE(VARYING\_STRING),intent(in)** *LABEL\_TYPE*, **INTEGER(INTG),intent(out)** *ERR*, **TYPE(VARYING\_STRING),intent(out)** *ERROR*, \*) [private]

Finding basis information.

**Parameters:**

**INTERPOLATION** xi interpolation type

**LABEL\_TYPE** label type

**ERR** The error code

**ERROR** The error string

Definition at line 1620 of file field\_IO\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    and    **BASE\_ROUTINES::EXITS()**.

Here is the call graph for this function:

**6.19.2.37 subroutine FIELD\_IO\_ROUTINES::STRING\_TO\_MUTI\_INTEGERS\_VS**  
**(TYPE(VARYING\_STRING),intent(in) STRING, INTEGER(INTG),intent(in)**  
**NUMBER\_OF\_INTEGERS, INTEGER(INTG),dimension(:),intent(inout) INTG\_DATA,**  
**INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,**  
 $\ast$ ) [private]

Definition at line 5706 of file field\_IO\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    and    **BASE\_ROUTINES::EXITS()**.

Here is the call graph for this function:

**6.19.2.38 subroutine FIELD\_IO\_ROUTINES::STRING\_TO\_MUTI\_REALS\_VS**  
**(TYPE(VARYING\_STRING),intent(in) STRING, INTEGER(INTG),intent(in)**  
**NUMBER\_OF\_REALS, REAL(DP),dimension(:),intent(inout) REAL\_DATA,**  
**INTEGER(INTG),intent(in) POSITION, INTEGER(INTG),intent(out) ERR,**  
**TYPE(VARYING\_STRING),intent(out) ERROR,  $\ast$ ) [private]**

Definition at line 5749 of file field\_IO\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    and    **BASE\_ROUTINES::EXITS()**.

Here is the call graph for this function:

### 6.19.3 Variable Documentation

**6.19.3.1 INTEGER(INTG),parameter FIELD\_IO\_ROUTINES::FIELD\_IO\_COMPONENT\_-  
**LABEL = 3****

Definition at line 78 of file field\_IO\_routines.f90.

**6.19.3.2 INTEGER(INTG),parameter FIELD\_IO\_ROUTINES::FIELD\_IO\_DERIVATIVE\_-  
**LABEL = 4****

Definition at line 79 of file field\_IO\_routines.f90.

**6.19.3.3 INTEGER(INTG),parameter FIELD\_IO\_ROUTINES::FIELD\_IO\_FIELD\_LABEL = 1**

Type for lable.

Definition at line 76 of file field\_IO\_routines.f90.

**6.19.3.4 INTEGER(INTG),parameter FIELD\_IO\_ROUTINES::FIELD\_IO\_SCALE\_-  
FACTORS\_NUMBER\_TYPE = 5**

Type of scale factor.

Definition at line 82 of file field\_IO\_routines.f90.

**6.19.3.5 INTEGER(INTG),parameter FIELD\_IO\_ROUTINES::FIELD\_IO\_SCALE\_-  
FACTORS\_PROPERTY\_TYPE = 6**

Definition at line 83 of file field\_IO\_routines.f90.

**6.19.3.6 INTEGER(INTG),parameter FIELD\_IO\_ROUTINES::FIELD\_IO\_VARIABLE\_LABEL  
= 2**

Definition at line 77 of file field\_IO\_routines.f90.

**6.19.3.7 INTEGER(INTG),parameter FIELD\_IO\_ROUTINES::SHAPE\_SIZE = 3**

size of shape

Definition at line 73 of file field\_IO\_routines.f90.

## 6.20 FIELD\_ROUTINES Namespace Reference

This module handles all field related routines.

### Classes

- interface [FIELD\\_COMPONENT\\_INTERPOLATION\\_SET](#)
- interface [FIELD\\_COMPONENT\\_MESH\\_COMPONENT\\_SET](#)
- interface [FIELD\\_DEPENDENT\\_TYPE\\_SET](#)
- interface [FIELD\\_DIMENSION\\_SET](#)
- interface [FIELD\\_GEOMETRIC\\_FIELD\\_SET](#)
- interface [FIELD\\_MESH\\_DECOMPOSITION\\_SET](#)
- interface [FIELD\\_NUMBER\\_OF\\_COMPONENTS\\_SET](#)
- interface [FIELD\\_NUMBER\\_OF\\_VARIABLES\\_SET](#)
- interface [FIELD\\_SCALING\\_TYPE\\_SET](#)
- interface [FIELD\\_TYPE\\_SET](#)

### Functions

- INTEGER(INTG) [FIELD\\_COMPONENT\\_INTERPOLATION\\_GET](#) (FIELD, FIELD VARIABLE\_NUMBER, FIELD\_COMPONENT\_NUMBER, ERR, ERROR)
 

*Gets the interpolation type for a field variable component identified by a pointer.*
- subroutine [FIELD\\_COMPONENT\\_INTER](#) (USER\_NUMBER, FIELD\_VARIABLE\_NUMBER, FIELD\_COMPONENT\_NUMBER, REGION,&INTERPOLATION\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the interpolation type for a field variable component identified by a user number and component number on a region.*
- subroutine [FI](#) (FIELD, FIELD\_VARIABLE\_NUMBER, FIELD\_COMPONENT\_NUMBER, INTERPOLATION\_TYPE,&ERR, ERROR,\*)
 

*Sets/changes the interpolation type for a field variable component identified by a pointer.*
- INTEGER(INTG) [FIELD\\_COMPONENT\\_MESH\\_COMPONENT\\_GET](#) (FIELD, FIELD VARIABLE\_NUMBER, FIELD\_COMPONENT\_NUMBER, ERR, ERROR)
 

*Gets the mesh component number for a field variable component identified by a pointer to a field and a field variable number.*
- subroutine [FIELD\\_COMPONENT\\_MESH\\_COM](#) (USER\_NUMBER, FIELD\_VARIABLE NUMBER, FIELD\_COMPONENT\_NUMBER, REGION,&MESH\_COMPONENT\_NUMBER, ERR, ERROR,\*)
 

*Sets/changes the mesh component number for a field variable component identified by a user number, component number and variable number on a region.*
- subroutine [FIELD\\_VARIABLE\\_COMPONENT\\_FINALISE](#) (FIELD VARIABLE COMPONENT, ERR, ERROR,\*)
 

*Finalises a field variable component and deallocates all memory.*
- subroutine [FIELD\\_VARIABLE\\_COMPONENT\\_INITIALISE](#) (FIELD, VARIABLE\_NUMBER, COMPONENT\_NUMBER, ERR, ERROR,\*)
 

*Initialises a field variable component.*

- subroutine **FIELD\_CREATE\_FINISH** (REGION, FIELD, ERR, ERROR,\*)
 

*Finishes the creation of a field on a region.*
- subroutine **FIELD\_CREATE\_VALUES\_CACHE\_FINALISE** (FIELD, ERR, ERROR,\*)
 

*Finalise the create values cache for a field.*
- subroutine **FIELD\_CREATE\_VALUES\_CACHE\_INITIALISE** (FIELD, ERR, ERROR,\*)
 

*Initialises the create values cache for a field.*
- INTEGER(INTG) **FIELD\_DEPENDENT\_TYPE\_GET** (FIELD, ERR, ERROR)
 

*Gets the dependent type for a field indentified by a pointer.*
- subroutine **FIELD\_DEPENDENT\_TYPE\_SET\_NUMBER** (USER\_NUMBER, REGION, DEPENDENT\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the dependent type for a field identified by a user number.*
- subroutine **FIELD\_DEPENDENT\_TYPE\_SET\_PTR** (FIELD, DEPENDENT\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the dependent type for a field indentified by a pointer.*
- subroutine **FIELD\_DESTROY** (FIELD, ERR, ERROR,\*)
 

*Destroys a field identified by a pointer to a field.*
- INTEGER(INTG) **FIELD\_DIMENSION\_GET** (FIELD, ERR, ERROR)
 

*Gets the field dimension for a field identified by a pointer.*
- subroutine **FIELD\_DIMENSION\_SET\_NUMBER** (USER\_NUMBER, REGION, FIELD\_DIMENSION, ERR, ERROR,\*)
 

*Sets/changes the field dimension for a field identified by a user number.*
- subroutine **FIELD\_DIMENSION\_SET\_PTR** (FIELD, FIELD\_DIMENSION, ERR, ERROR,\*)
 

*Sets/changes the field dimension for a field indentified by a pointer.*
- subroutine **FIELD\_INTERPOLATE\_GAUSS** (PARTIAL\_DERIVATIVE\_TYPE, QUADRATURE\_SCHEME, GAUSS\_POINT\_NUMBER, INTERPOLATED\_POINT, ERR, ERROR,\*)
 

*Interpolates a field at a gauss point to give an interpolated point. PARTIAL\_DERIVATIVE\_TYPE controls which partial derivatives are evaluated. If it is NO\_PART\_DERIV then only the field values are interpolated. If it is FIRST\_PART\_DERIV then the field values and first partial derivatives are interpolated. If it is SECOND\_PART\_DERIV the the field values and first and second partial derivatives are evaluated. Old CMISS name XEXG, ZEXG.*
- subroutine **FIELD\_INTERPOLATE\_XI** (PARTIAL\_DERIVATIVE\_TYPE, XI, INTERPOLATED\_POINT, ERR, ERROR,\*)
 

*Interpolates a field at a xi location to give an interpolated point. XI is the element location to be interpolated at. PARTIAL\_DERIVATIVE\_TYPE controls which partial derivatives are evaluated. If it is NO\_PART\_DERIV then only the field values are interpolated. If it is FIRST\_PART\_DERIV then the field values and first partial derivatives are interpolated. If it is SECOND\_PART\_DERIV the the field values and first and second partial derivatives are evaluated. Old CMISS name PXI.*

- subroutine [FIELD\\_INTERPOLATED\\_POINT\\_FINALISE](#) (INTERPOLATED\_POINT, ERR, ERROR,\*)
 

*Finalises the interpolated point and deallocates all memory.*
- subroutine [FIELD\\_INTERPOLATED\\_POINT\\_INITIALISE](#) (INTERPOLATION\_PARAMETERS, INTERPOLATED\_POINT, ERR, ERROR,\*)
 

*Initialises the interpolated point for an interpolation parameters.*
- subroutine [FIELD\\_INTERPOLATED\\_POINT\\_METRICS\\_CALCULATE](#) (JACOBIAN\_TYPE, INTERPOLATED\_POINT\_METRICS, ERR, ERROR,\*)
 

*Calculates the interpolated point metrics and the associated interpolated point.*
- subroutine [FIELD\\_INTERPOLATED\\_POINT\\_METRICS\\_FINALISE](#) (INTERPOLATED\_POINT\_METRICS, ERR, ERROR,\*)
 

*Finalises the interpolated point metrics and deallocate all memory.*
- subroutine [FIELD\\_INTERPOLATED\\_POINT\\_METRICS\\_INITIALISE](#) (INTERPOLATED\_POINT, INTERPOLATED\_POINT\_METRICS, ERR, ERROR,\*)
 

*Initialises the interpolated point metrics for an interpolated point.*
- subroutine [FIELD\\_INTERPOLATION\\_PARAMETERS\\_ELEMENT\\_GET](#) (PARAMETER\_SET\_NUMBER, ELEMENT\_NUMBER, INTERPOLATION\_PARAMETERS, ERR, ERROR,\*)
 

*Gets the interpolation parameters for a particular element. Old [CMISS](#) name XPXE, ZPZE.*
- subroutine [FIELD\\_INTERPOLATION\\_PARAMETERS\\_FINALISE](#) (INTERPOLATION\_PARAMETERS, ERR, ERROR,\*)
 

*Finalises the interpolation parameters and deallocate all memory.*
- subroutine [FIELD\\_INTERPOLATION\\_PARAMETERS\\_INITIALISE](#) (FIELD, VARIABLE\_NUMBER, INTERPOLATION\_PARAMETERS, ERR, ERROR,\*)
 

*Initialises the interpolation parameters for a field variable.*
- subroutine [FIELD\\_INTERPOLATION\\_PARAMETERS\\_LINE\\_GET](#) (PARAMETER\_SET\_NUMBER, LINE\_NUMBER, INTERPOLATION\_PARAMETERS, ERR, ERROR,\*)
 

*Gets the interpolation parameters for a particular line. Old [CMISS](#) name XPXE, ZPZE.*
- subroutine [FIELD\\_VARIABLES\\_FINALISE](#) (FIELD, ERR, ERROR,\*)
 

*Finalises the field variables for a field and deallocate all memory.*
- subroutine [FIELD\\_VARIABLES\\_INITIALISE](#) (FIELD, ERR, ERROR,\*)
 

*Initialises the field variables.*
- subroutine [FIELDS\\_FINALISE](#) (REGION, ERR, ERROR,\*)
 

*Finalises the fields for the given region and deallocate all memory.*
- subroutine [FIELDS\\_INITIALISE](#) (REGION, ERR, ERROR,\*)
 

*Initialises the fields for the given region.*

## Variables

- INTEGER(INTG), parameter **FIELD\_INDEPENDENT\_TYPE** = 1  
*Independent field type.*
- INTEGER(INTG), parameter **FIELD\_DEPENDENT\_TYPE** = 2  
*Dependent field type.*
- INTEGER(INTG), parameter **FIELD\_SCALAR\_DIMENSION\_TYPE** = 1  
*Scalar field.*
- INTEGER(INTG), parameter **FIELD\_VECTOR\_DIMENSION\_TYPE** = 2  
*Vector field.*
- INTEGER(INTG), parameter **FIELD\_TENSOR\_DIMENSION\_TYPE** = 2  
*Tensor field.*
- INTEGER(INTG), parameter **FIELD\_GEOMETRIC\_TYPE** = 1  
*Geometric field.*
- INTEGER(INTG), parameter **FIELD\_FIBRE\_TYPE** = 2  
*Fibre field.*
- INTEGER(INTG), parameter **FIELD\_GENERAL\_TYPE** = 3  
*General field.*
- INTEGER(INTG), parameter **FIELD\_MATERIAL\_TYPE** = 4  
*Material field.*
- INTEGER(INTG), parameter **FIELD\_CONSTANT\_INTERPOLATION** = 1  
*Constant interpolation. One parameter for the field.*
- INTEGER(INTG), parameter **FIELD\_ELEMENT\_BASED\_INTERPOLATION** = 2  
*Element based interpolation. Parameters are different in each element.*
- INTEGER(INTG), parameter **FIELD\_NODE\_BASED\_INTERPOLATION** = 3  
*Node based interpolation. Parameters are nodal based and a basis function is used.*
- INTEGER(INTG), parameter **FIELD\_GRID\_POINT\_BASED\_INTERPOLATION** = 4  
*Grid point based interpolation. Parameters are different at each grid point.*
- INTEGER(INTG), parameter **FIELD\_GAUSS\_POINT\_BASED\_INTERPOLATION** = 5  
*Gauss point based interpolation. Parameters are different at each Gauss point.*
- INTEGER(INTG), parameter **FIELD\_NUMBER\_OF\_VARIABLE\_TYPES** = 4  
*Number of different field variable types possible.*
- INTEGER(INTG), parameter **FIELD\_STANDARD\_VARIABLE\_TYPE** = 1  
*Standard variable type i.e.,  $u$ .*

- INTEGER(INTG), parameter **FIELD\_NORMAL\_VARIABLE\_TYPE** = 2  
*Normal derivative variable type i.e.,  $du/dn$ .*
- INTEGER(INTG), parameter **FIELD\_TIME\_DERIV1\_VARIABLE\_TYPE** = 3  
*First time derivative variable type i.e.,  $du/dt$ .*
- INTEGER(INTG), parameter **FIELD\_TIME\_DERIV2\_VARIABLE\_TYPE** = 4  
*Second type derivative variable type i.e.,  $d^2u/dt^2$ .*
- INTEGER(INTG), parameter **FIELD\_CONSTANT\_DOF\_TYPE** = 1  
*The dof is from a field variable component with constant interpolation.*
- INTEGER(INTG), parameter **FIELD\_ELEMENT\_DOF\_TYPE** = 2  
*The dof is from a field variable component with element based interpolation.*
- INTEGER(INTG), parameter **FIELD\_NODE\_DOF\_TYPE** = 3  
*The dof is from a field variable component with node based interpolation.*
- INTEGER(INTG), parameter **FIELD\_POINT\_DOF\_TYPE** = 4  
*The dof is from a field variable component with point based interpolation.*
- INTEGER(INTG), parameter **FIELD\_NUMBER\_OF\_SET\_TYPES** = 99  
*The maximum number of different parameter sets for a field.*
- INTEGER(INTG), parameter **FIELD\_VALUES\_SET\_TYPE** = 1  
*The parameter set corresponding to the field values.*
- INTEGER(INTG), parameter **FIELD\_BOUNDARY\_CONDITIONS\_SET\_TYPE** = 2  
*The parameter set corresponding to the field boundary conditions.*
- INTEGER(INTG), parameter **FIELD\_INITIAL\_CONDITIONS\_SET\_TYPE** = 3  
*The parameter set corresponding to the field initial conditions.*
- INTEGER(INTG), parameter **FIELD\_ANALYTIC\_SET\_TYPE** = 4  
*The parameter set corresponding to the field values.*
- INTEGER(INTG), parameter **FIELD\_NO\_SCALING** = 0  
*The field is not scaled.*
- INTEGER(INTG), parameter **FIELD\_UNIT\_SCALING** = 1  
*The field has unit scaling.*
- INTEGER(INTG), parameter **FIELD\_ARC\_LENGTH\_SCALING** = 2  
*The field has arc length scaling.*
- INTEGER(INTG), parameter **FIELD\_ARITHMETIC\_MEAN\_SCALING** = 3  
*The field has arithmetic mean of the arc length scaling.*
- INTEGER(INTG), parameter **FIELD\_HARMONIC\_MEAN\_SCALING** = 4  
*The field has geometric mean of the arc length scaling.*

### 6.20.1 Detailed Description

This module handles all field related routines.

### 6.20.2 Function Documentation

**6.20.2.1 subroutine FIELD\_ROUTINES::FI** (TYPE(FIELD\_TYPE),pointer  
**FIELD**, INTEGER(INTG),intent(in) **FIELD\_VARIABLE\_NUMBER**,  
 INTEGER(INTG),intent(in) **FIELD\_COMPONENT\_NUMBER**,  
 INTEGER(INTG),intent(in) **INTERPOLATION\_TYPE**, &,intent(out) **ERR**,  
 INTEGER(INTG),intent(out) **error**, \*) [private]

Sets/changes the interpolation type for a field variable component identified by a pointer.

Sets/changes the mesh component number for a field variable component identified by a pointer to a field and a field variable number.

**Parameters:**

**FIELD** A pointer to the field to set the interpolation for

**FIELD\_VARIABLE\_NUMBER** The field variable number of the field variable component to set

**FIELD\_COMPONENT\_NUMBER** The field component number of the field variable component to set

**INTERPOLATION\_TYPE** The interpolation type to set

**See also:**

[FIELD\\_ROUTINES::VariableTypes](#),[FIELD\\_ROUTINES](#)

Definition at line 358 of file field\_routines.f90.

References [BASE\\_ROUTINES::ENTERS\(\)](#), [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::EXITS\(\)](#), [FIELD\\_CONSTANT\\_INTERPOLATION](#), [FIELD\\_ELEMENT\\_BASED\\_INTERPOLATION](#), [FIELD\\_GAUSS\\_POINT\\_BASED\\_INTERPOLATION](#), [FIELD\\_GRID\\_POINT\\_BASED\\_INTERPOLATION](#), and [FIELD\\_NODE\\_BASED\\_INTERPOLATION](#).

Here is the call graph for this function:

**6.20.2.2 subroutine FIELD\_ROUTINES::FIELD\_COMPONENT\_INTER**  
 (INTEGER(INTG),intent(in) **USER\_NUMBER**, INTEGER(INTG),intent(in) **FIELD\_VARIABLE\_NUMBER**, INTEGER(INTG),intent(in) **FIELD\_COMPONENT\_NUMBER**,  
 TYPE(REGION\_TYPE),pointer **REGION**, &,intent(in) **INTERPOLATION\_TYPE**,  
 INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)  
 [private]

Sets/changes the interpolation type for a field variable component identified by a user number and component number on a region.

**Parameters:**

**FIELD\_VARIABLE\_NUMBER** The field variable number of the field variable component

**FIELD\_COMPONENT\_NUMBER** The field component number of the field variable component

**REGION** The region containing the field

**ERR** The error code

**ERROR** The error string

Definition at line 323 of file field\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.20.2.3 INTEGER(INTG) FIELD\_ROUTINES::FIELD\_COMPONENT\_INTERPOLATION\_GET (TYPE(FIELD\_TYPE),pointer *FIELD*, INTEGER(INTG),intent(in) *FIELD\_VARIABLE\_NUMBER*, INTEGER(INTG),intent(in) *FIELD\_COMPONENT\_NUMBER*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*)**

Gets the interpolation type for a field variable component identified by a pointer.

#### Parameters:

***FIELD*** A pointer to the field to set the interpolation for

***FIELD\_VARIABLE\_NUMBER*** The field variable number of the field variable component to set

***FIELD\_COMPONENT\_NUMBER*** The field component number of the field variable component to set

***ERR*** The error code

***ERROR*** The error string

Definition at line 274 of file field\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.20.2.4 subroutine FIELD\_ROUTINES::FIELD\_COMPONENT\_MESH\_COM (INTEGER(INTG),intent(in) *USER\_NUMBER*, INTEGER(INTG),intent(in) *FIELD\_VARIABLE\_NUMBER*, INTEGER(INTG),intent(in) *FIELD\_COMPONENT\_NUMBER*, TYPE(REGION\_TYPE),pointer *REGION*, &,intent(in) *MESH\_COMPONENT\_NUMBER*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Sets/changes the mesh component number for a field variable component identified by a user number, component number and variable number on a region.

#### Todo

change variable number to variable type.

#### Parameters:

***USER\_NUMBER*** The user number of the field to set the mesh component for

***FIELD\_VARIABLE\_NUMBER*** The field variable number of the field variable component to set

***FIELD\_COMPONENT\_NUMBER*** The field component number of the field variable component to set

**REGION** A pointer to the region containing the field

**ERR** The error code

**ERROR** The error string

Definition at line 470 of file field\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `FIELD_CONSTANT_INTERPOLATION`,    `FIELD_ELEMENT_BASED_INTERPOLATION`,    `FIELD_GAUSS_POINT_BASED_INTERPOLATION`,    `FIELD_GRID_POINT_BASED_INTERPOLATION`, and `FIELD_NODE_BASED_INTERPOLATION`.

Here is the call graph for this function:

```
6.20.2.5 INTEGER(INTG) FIELD_ROUTINES::FIELD_COMPONENT_
MESH_COMPONENT_GET (TYPE(FIELD_TYPE),pointer
FIELD, INTEGER(INTG),intent(in) FIELD_VARIABLE_NUMBER,
INTEGER(INTG),intent(in) FIELD_COMPONENT_NUMBER,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)
[private]
```

Gets the mesh component number for a field variable component identified by a pointer to a field and a field variable number.

#### Parameters:

**FIELD** A pointer to the field to set the mesh component for

**FIELD\_VARIABLE\_NUMBER** The field variable number to set the field variable component for

**FIELD\_COMPONENT\_NUMBER** The field component number to set the field variable component for

**ERR** The error code

**ERROR** The error string

Definition at line 421 of file field\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.20.2.6 subroutine FIELD_ROUTINES::FIELD_CREATE_FINISH (TYPE(REGION_-
TYPE),pointer REGION, TYPE(FIELD_TYPE),pointer FIELD,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)
```

Finishes the creation of a field on a region.

#### Parameters:

**REGION** A pointer to the region containing the field

**FIELD** A pointer to the field to finish the creation of

**ERR** The error code

**ERROR** The error string

Definition at line 879 of file field\_routines.f90.

References            `BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE,            BASE_-ROUTINES::DIAGNOSTICS1,    BASE_ROUTINES::ENTERS(),    BASE_ROUTINES::ERRORS(),    BASE_ROUTINES::EXITS(),    FIELD_ARITHMETIC_MEAN_SCALING,    FIELD_CREATE_VALUES_CACHE_FINALISE(),    FIELD_CREATE_VALUES_CACHE_INITIALISE(),    FIELD_-GEOMETRIC_TYPE,    FIELD_INDEPENDENT_TYPE,    FIELD_NUMBER_OF_VARIABLE_TYPES,    FIELD_VALUES_SET_TYPE,    FIELD_VARIABLES_INITIALISE(),    FIELD_VECTOR_-DIMENSION_TYPE, and KINDS::PTR.`

Referenced by `FIELD_IO_ROUTINES::FIELD_IO_CREATE_FIELDS(),    FINITE_ELASTICITY_-ROUTINES::FINITE_ELASTICITY_EQUATIONS_SET_SETUP(),    FINITE_ELASTICITY_-ROUTINES::FINITE_ELASTICITYFINITE_ELEMENT_RESIDUAL_EVALUATE(),    LAPLACE_-EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), and LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_EQUATIONS_SET_SETUP()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### **6.20.2.7 subroutine FIELD\_ROUTINES::FIELD\_CREATE\_VALUES\_CACHE\_FINALISE (TYPE(FIELD\_TYPE),pointer FIELD, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Finalise the create values cache for a field.

##### **Parameters:**

**FIELD** A pointer to the field to finalise the create values cache for

**ERR** The error code

**ERROR** The error string

Definition at line 1060 of file field\_routines.f90.

References    `BASE_ROUTINES::ENTERS(),    BASE_ROUTINES::ERRORS(),    and    BASE_-ROUTINES::EXITS()`.

Referenced by `FIELD_CREATE_FINISH(), and FIELD_DESTROY()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### **6.20.2.8 subroutine FIELD\_ROUTINES::FIELD\_CREATE\_VALUES\_CACHE\_INITIALISE (TYPE(FIELD\_TYPE),pointer FIELD, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Initialises the create values cache for a field.

##### **Parameters:**

**FIELD** A pointer to the field to initialise the create values cache for

**ERR** The error code

**ERROR** The error string

Definition at line 1093 of file field\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `FIELD_FIBRE_TYPE`,    `FIELD_GENERAL_TYPE`,    `FIELD_GEOMETRIC_TYPE`,    `FIELD_MATERIAL_TYPE`, and `FIELD_NODE_BASED_INTERPOLATION`.

Referenced by `FIELD_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### **6.20.2.9 INTEGER(INTG) FIELD\_ROUTINES::FIELD\_DEPENDENT\_TYPE\_GET (TYPE(FIELD\_TYPE),pointer *FIELD*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*)**

Gets the dependent type for a field indentified by a pointer.

**Parameters:**

***FIELD*** A pointer to the field to set/change the dependent type for

***ERR*** The error code

***ERROR*** The error string

Definition at line 1153 of file `field_routines.f90`.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

#### **6.20.2.10 subroutine FIELD\_ROUTINES::FIELD\_DEPENDENT\_TYPE\_SET\_NUMBER (INTEGER(INTG),intent(in) *USER\_NUMBER*, TYPE(REGION\_TYPE),pointer *REGION*, INTEGER(INTG),intent(in) *DEPENDENT\_TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Sets/changes the dependent type for a field identified by a user number.

**Parameters:**

***USER\_NUMBER*** The user number of the field to set the dependent type for

***REGION*** A pointer to the region containing the field

***DEPENDENT\_TYPE*** The dependent type to set/change for the field

**See also:**

[FIELD\\_ROUTINES::DependentTypes](#), [FIELD\\_ROUTINES](#)

***ERR*** The error code

***ERROR*** The error string

Definition at line 1183 of file `field_routines.f90`.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `FIELD_DEPENDENT_TYPE_SET_PTR()`.

Here is the call graph for this function:

**6.20.2.11 subroutine FIELD\_ROUTINES::FIELD\_DEPENDENT\_TYPE\_SET\_PTR  
 $(TYPE(FIELD\_TYPE),pointer FIELD, INTEGER(INTG),intent(in) DEPENDENT\_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, *)$  [private]**

Sets/changes the dependent type for a field indentified by a pointer.

**Parameters:**

**FIELD** A pointer to the field to set/change the dependent type for

**DEPENDENT\_TYPE** The dependent type to set/change

See also:

[FIELD\\_ROUTINES::DependentTypes](#), [FIELD\\_ROUTINES](#)

**ERR** The error code

**ERROR** The error string

Definition at line 1214 of file field\_routines.f90.

References [BASE\\_ROUTINES::ENTERS\(\)](#), [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::EXITS\(\)](#), [FIELD\\_DEPENDENT\\_TYPE](#), and [FIELD\\_INDEPENDENT\\_TYPE](#).

Referenced by [FIELD\\_DEPENDENT\\_TYPE\\_SET\\_NUMBER\(\)](#).

Here is the call graph for this function:

Here is the caller graph for this function:

**6.20.2.12 subroutine FIELD\_ROUTINES::FIELD\_DESTROY  
 $(TYPE(FIELD\_TYPE),pointer FIELD, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, *)$**

Destroys a field identified by a pointer to a field.

**Parameters:**

**FIELD** A pointer to the field to destroy

**ERR** The error code

**ERROR** The error string

Definition at line 1289 of file field\_routines.f90.

References [BASE\\_ROUTINES::ENTERS\(\)](#), [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::EXITS\(\)](#), [FIELD\\_CREATE\\_VALUES\\_CACHE\\_FINALISE\(\)](#), [FIELD\\_VARIABLES\\_FINALISE\(\)](#), and [KINDS::PTR](#).

Referenced by [FIELDS\\_FINALISE\(\)](#).

Here is the call graph for this function:

Here is the caller graph for this function:

**6.20.2.13 INTEGER(INTG) FIELD\_ROUTINES::FIELD\_DIMENSION\_GET  
 $(TYPE(FIELD\_TYPE),pointer FIELD, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR)$  [private]**

Gets the field dimension for a field indentified by a pointer.

**Parameters:**

***FIELD*** A pointer to the field to set/change the dimension for  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 1391 of file field\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.20.2.14 subroutine `FIELD_ROUTINES::FIELD_DIMENSION_SET_NUMBER`**  
`(INTEGER(INTG),intent(in) USER_NUMBER, TYPE(REGION_TYPE),pointer REGION, INTEGER(INTG),intent(in) FIELD_DIMENSION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Sets/changes the field dimension for a field identified by a user number.

**Parameters:**

***USER\_NUMBER*** The user number of the field to set the dimension for  
***REGION*** A pointer to the region containing the field  
***FIELD\_DIMENSION*** The dimension to set/change

**See also:**

[FIELD\\_ROUTINES::DimensionTypes](#), [FIELD\\_ROUTINES](#)

***ERR*** The error code

***ERROR*** The error string

Definition at line 1421 of file field\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `FIELD_DIMENSION_SET_PTR()`.

Here is the call graph for this function:

**6.20.2.15 subroutine `FIELD_ROUTINES::FIELD_DIMENSION_SET_PTR`** (`TYPE(FIELD_TYPE),pointer FIELD, INTEGER(INTG),intent(in) FIELD_DIMENSION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`)

Sets/changes the field dimension for a field identified by a pointer.

**Parameters:**

***FIELD*** A pointer to the field to set/change the dimension for  
***FIELD\_DIMENSION*** The field dimension to set/change

**See also:**

[FIELD\\_ROUTINES::DimensionTypes](#), [FIELD\\_ROUTINES](#)

***ERR*** The error code

**ERROR** The error string

Definition at line 1452 of file field\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_SCALAR_DIMENSION_TYPE`, and `FIELD_VECTOR_DIMENSION_TYPE`.

Referenced by `FIELD_DIMENSION_SET_NUMBER()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.20.2.16 subroutine FIELD\_ROUTINES::FIELD\_INTERPOLATE\_GAUSS**  
`(INTEGER(INTG),intent(in) PARTIAL_DERIVATIVE_TYPE,`  
`INTEGER(INTG),intent(in) QUADRATURE_SCHEME, INTEGER(INTG),intent(in)`  
`GAUSS_POINT_NUMBER, TYPE(FIELD_INTERPOLATED_POINT_TYPE),pointer`  
`INTERPOLATED_POINT, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Interpolates a field at a gauss point to give an interpolated point. `PARTIAL_DERIVATIVE_TYPE` controls which partial derivatives are evaluated. If it is `NO_PART_DERIV` then only the field values are interpolated. If it is `FIRST_PART_DERIV` then the field values and first partial derivatives are interpolated. If it is `SECOND_PART_DERIV` the the field values and first and second partial derivatives are evaluated. Old CMISS name XEXG, ZEXG.

#### Parameters:

**PARTIAL\_DERIVATIVE\_TYPE** The partial derivative type of the provided field interpolation

**QUADRATURE\_SCHEME** The quadrature scheme of the Gauss points

#### See also:

`BASIS_ROUTINES_QuadratureSchemes,BASIS_ROUTINES`

**GAUSS\_POINT\_NUMBER** The number of the Gauss point to interpolate the field at

**INTERPOLATED\_POINT** The pointer to the interpolated point which will contain the field interpolation information at the specified Gauss point

**ERR** The error code

**ERROR** The error string

Definition at line 1524 of file field\_routines.f90.

References `COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST()`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `KINDS::PTR`.

Referenced by `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATIONFINITEELEMENTCALCULATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.20.2.17 subroutine FIELD\_ROUTINES::FIELD\_INTERPOLATE\_XI**  
**(INTEGER(INTG),intent(in) PARTIAL\_DERIVATIVE\_TYPE,**  
**REAL(DP),dimension(:),intent(in) XI, TYPE(FIELD\_INTERPOLATED\_POINT\_-**  
**TYPE),pointer INTERPOLATED\_POINT, INTEGER(INTG),intent(out) ERR,**  
**TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Interpolates a field at a xi location to give an interpolated point. XI is the element location to be interpolated at. PARTIAL\_DERIVATIVE\_TYPE controls which partial derivatives are evaluated. If it is NO\_PART\_DERIV then only the field values are interpolated. If it is FIRST\_PART\_DERIV then the field values and first partial derivatives are interpolated. If it is SECOND\_PART\_DERIV the the field values and first and second partial derivatives are evaluated. Old CMISS name PXI.

**Parameters:**

**PARTIAL\_DERIVATIVE\_TYPE** The partial derivative type of the provide field interpolation  
**XI** XI(ni). The ni'th Xi coordinate to evaluate the field at  
**INTERPOLATED\_POINT** The pointer to the interpolated point which will contain the field interpolation information at the specified Xi point  
**ERR** The error code  
**ERROR** The error string

Definition at line 1609 of file field\_routines.f90.

References COORDINATE\_ROUTINES::COORDINATE\_INTERPOLATION\_ADJUST(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and KINDS::PTR.

Referenced by FIELD\_INTERPOLATION\_PARAMETERS\_LINE\_GET(), FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITYFINITE\_ELEMENT\_RESIDUAL\_EVALUATE(), and LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_INTE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.20.2.18 subroutine FIELD\_ROUTINES::FIELD\_INTERPOLATED\_POINT\_FINALISE**  
**(TYPE(FIELD\_INTERPOLATED\_POINT\_TYPE),pointer INTERPOLATED\_POINT,**  
**INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,**  
**\*)**

Finalises the interpolated point and deallocates all memory.

**Parameters:**

**INTERPOLATED\_POINT** A pointer to the interpolated point to finalise  
**ERR** The error code  
**ERROR** The error string

Definition at line 1704 of file field\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by FIELD\_INTERPOLATED\_POINT\_INITIALISE(), FIELD\_INTERPOLATION\_PARAMETERS\_LINE\_GET(), and LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_ANALYTIC\_CALCULATE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.20.2.19 subroutine FIELD\_ROUTINES::FIELD\_INTERPOLATED\_POINT\_INITIALISE**  
`(TYPE(FIELD_INTERPOLATION_PARAMETERS_TYPE),pointer  
INTERPOLATION_PARAMETERS, TYPE(FIELD_INTERPOLATED_POINT_-  
TYPE),pointer INTERPOLATED_POINT, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING_STRING),intent(out) ERROR, *)`

Initialises the interpolated point for an interpolation parameters.

**Parameters:**

**INTERPOLATION\_PARAMETERS** A pointer to the interpolation parameters to initialise the interpolated point for

**INTERPOLATED\_POINT** On exit, A pointer to the interpolated point that has been initialised

**ERR** The error code

**ERROR** The error string

Definition at line 1731 of file field\_routines.f90.

References KINDS::\_DP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and FIELD\_INTERPOLATED\_POINT\_FINALISE().

Referenced by FIELD\_INTERPOLATION\_PARAMETERS\_LINE\_GET(), FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITYFINITE\_ELEMENT\_RESIDUAL\_EVALUATE(), and LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_INTE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.20.2.20 subroutine FIELD\_ROUTINES::FIELD\_INTERPOLATED\_POINT\_-  
METRICS\_CALCULATE (INTEGER(INTG),intent(in) JACOBIAN\_TYPE,  
TYPE(FIELD\_INTERPOLATED\_POINT\_METRICS\_TYPE),pointer  
INTERPOLATED\_POINT\_METRICS, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Calculates the interpolated point metrics and the associated interpolated point.

**Parameters:**

**JACOBIAN\_TYPE** The Jacobian type of the calculation

**See also:**

COORDINATE\_ROUTINES\_JacobianTypes, [COORDINATE\\_ROUTINES](#)

**INTERPOLATED\_POINT\_METRICS** A pointer to the interpolated point metrics

**ERR** The error code

**ERROR** The error string

Definition at line 1779 of file field\_routines.f90.

References COORDINATE\_ROUTINES::COORDINATE\_METRICS\_CALCULATE(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), FIELD\_FIBRE\_TYPE, and FIELD\_GEOMETRIC\_TYPE.

Referenced by LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATIONFINITE\_ELEMENT\_Calculate().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.20.2.21 subroutine FIELD\_ROUTINES::FIELD\_INTERPOLATED\_POINT\_METRICS\_FINALISE (TYPE(FIELD\_INTERPOLATED\_POINT\_METRICS\_TYPE),pointer INTERPOLATED\_POINT\_METRICS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Finalises the interpolated point metrics and deallocates all memory.

#### Parameters:

**INTERPOLATED\_POINT\_METRICS** A pointer to the interpolated point metrics to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 1820 of file field\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by FIELD\_INTERPOLATED\_POINT\_METRICS\_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.20.2.22 subroutine FIELD\_ROUTINES::FIELD\_INTERPOLATED\_POINT\_METRICS\_INITIALISE (TYPE(FIELD\_INTERPOLATED\_POINT\_TYPE),pointer INTERPOLATED\_POINT, TYPE(FIELD\_INTERPOLATED\_POINT\_METRICS\_TYPE),pointer INTERPOLATED\_POINT\_METRICS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Initialises the interpolated point metrics for an interpolated point.

#### Parameters:

**INTERPOLATED\_POINT\_METRICS** On exit, a pointer to the interpolated point metrics that have been initialised

**ERR** The error code

**ERROR** The error string

Definition at line 1850 of file field\_routines.f90.

References KINDS::\_DP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and FIELD\_INTERPOLATED\_POINT\_METRICS\_FINALISE().

Here is the call graph for this function:

---

**6.20.2.23 subroutine FIELD\_ROUTINES::FIELD\_INTERPOLATION\_-  
PARAMETERS\_ELEMENT\_GET (INTEGER(INTG),intent(in)  
PARAMETER\_SET\_NUMBER, INTEGER(INTG),intent(in) ELEMENT\_NUMBER,  
TYPE(FIELD\_INTERPOLATION\_PARAMETERS\_TYPE),pointer  
INTERPOLATION\_PARAMETERS, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Gets the interpolation parameters for a particular element. Old [CMISS](#) name XPXE, ZPZE.

**Parameters:**

**PARAMETER\_SET\_NUMBER** The field parameter set number to get the element parameters for  
**ELEMENT\_NUMBER** The element number to get the element parameters for  
**INTERPOLATION\_PARAMETERS** A pointer to the interpolation parameters  
**ERR** The error code  
**ERROR** The error string

Definition at line 1914 of file field\_routines.f90.

References COORDINATE\_ROUTINES::COORDINATE\_INTERPOLATION\_-  
 PARAMETERS\_ADJUST(), BASE\_ROUTINES::DIAGNOSTIC\_OUTPUT\_TYPE, BASE\_-  
 ROUTINES::DIAGNOSTICS1, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(),  
 BASE\_ROUTINES::EXITS(), FIELD\_ARC\_LENGTH\_SCALING, FIELD\_ARITHMETIC\_-  
 MEAN\_SCALING, FIELD\_CONSTANT\_INTERPOLATION, FIELD\_ELEMENT\_BASED\_-  
 INTERPOLATION, FIELD\_GAUSS\_POINT\_BASED\_INTERPOLATION, FIELD\_GRID\_POINT\_-  
 BASED\_INTERPOLATION, FIELD\_HARMONIC\_MEAN\_SCALING, FIELD\_NO\_SCALING,  
 FIELD\_NODE\_BASED\_INTERPOLATION, FIELD\_UNIT\_SCALING, and KINDS::PTR.

Referenced by FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITYFINITE\_ELEMENT\_-  
 RESIDUAL\_EVALUATE(), LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_-  
 FINITE\_ELEMENT\_CALCULATE(), and LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_-  
 EQUATION\_INTE().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.20.2.24 subroutine FIELD\_ROUTINES::FIELD\_INTERPOLATION\_PARAMETERS\_-  
FINALISE (TYPE(FIELD\_INTERPOLATION\_PARAMETERS\_TYPE),pointer  
INTERPOLATION\_PARAMETERS, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Finalises the interpolation parameters and deallocates all memory.

**Parameters:**

**INTERPOLATION\_PARAMETERS** A pointer to the interpolation parameters to finalise  
**ERR** The error code  
**ERROR** The error string

Definition at line 2070 of file field\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-  
 ROUTINES::EXITS().

Referenced by FIELD\_INTERPOLATION\_PARAMETERS\_INITIALISE(), FIELD\_INTERPOLATION\_PARAMETERS\_LINE\_GET(), and LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_INTE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.20.2.25 subroutine FIELD\_ROUTINES::FIELD\_INTERPOLATION\_PARAMETERS\_INITIALISE** (TYPE(FIELD\_TYPE),pointer **FIELD**, INTEGER(INTG),intent(in) **VARIABLE\_NUMBER**, TYPE(FIELD\_INTERPOLATION\_PARAMETERS\_TYPE),pointer **INTERPOLATION\_PARAMETERS**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

Initialises the interpolation parameters for a field variable.

#### Parameters:

**FIELD** A pointer to the field to initialise the interpolation parameters for

**VARIABLE\_NUMBER** The field variable number to initialise the interpolation parameters for

**INTERPOLATION\_PARAMETERS** On exit, a pointer to the initialised interpolation parameters.

**ERR** The error code

**ERROR** The error string

Definition at line 2099 of file field\_routines.f90.

References KINDS::\_DP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), FIELD\_INTERPOLATION\_PARAMETERS\_FINALISE(), FIELD\_NUMBER\_OF\_VARIABLE\_TYPES, and KINDS::PTR.

Referenced by FIELD\_INTERPOLATION\_PARAMETERS\_LINE\_GET(), FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITYFINITE\_ELEMENT\_RESIDUAL\_EVALUATE(), and LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_INTE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.20.2.26 subroutine FIELD\_ROUTINES::FIELD\_INTERPOLATION\_PARAMETERS\_LINE\_GET** (INTEGER(INTG),intent(in) **PARAMETER\_SET\_NUMBER**, INTEGER(INTG),intent(in) **LINE\_NUMBER**, TYPE(FIELD\_INTERPOLATION\_PARAMETERS\_TYPE),pointer **INTERPOLATION\_PARAMETERS**, INTEGER(INTG),intent(out),parameter **ERR**, TYPE(VARYING\_STRING),intent(out),parameter **ERROR**, \*)

Gets the interpolation parameters for a particular line. Old CMISS name XPXE, ZPZE.

#### Parameters:

**PARAMETER\_SET\_NUMBER** The field parameter set number to get the line parameters for

**LINE\_NUMBER** The line number to get the line parameters for

**INTERPOLATION\_PARAMETERS** A pointer to the interpolation parameters

**ERR** The error code

**ERROR** The error string

Definition at line 2171 of file field\_routines.f90.

References KINDS::DP, COMP\_ENVIRONMENT::COMPUTATIONAL\_NODE\_NUMBER\_GET(), COMP\_ENVIRONMENT::COMPUTATIONAL\_NODES\_NUMBER\_GET(), COORDINATE\_ROUTINES::COORDINATE\_DERIVATIVE\_NORM(), COORDINATE\_ROUTINES::COORDINATE\_INTERPOLATION\_PARAMETERS\_ADJUST(), BASE\_ROUTINES::DIAGNOSTIC\_OUTPUT\_TYPE, BASE\_ROUTINES::DIAGNOSTICS1, BASE\_ROUTINES::DIAGNOSTICS2, DOMAIN\_MAPPINGS::DOMAIN\_LOCAL\_GHOST, DOMAIN\_MAPPINGS::DOMAIN\_LOCAL\_INTERNAL, DOMAIN\_MAPPINGS::DOMAIN\_MAPPINGS\_LOCAL\_FROM\_GLOBAL\_CALCULATE(), DOMAIN\_MAPPINGS::DOMAIN\_MAPPINGS\_MAPPING\_FINALISE(), DOMAIN\_MAPPINGS::DOMAIN\_MAPPINGS\_MAPPING\_GLOBAL\_INITIALISE(), DOMAIN\_MAPPINGS::DOMAIN\_MAPPINGS\_MAPPING\_INITIALISE(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), FIELD\_ARC\_LENGTH\_SCALING, FIELD\_ARITHMETIC\_MEAN\_SCALING, FIELD\_CONSTANT\_DOF\_TYPE, FIELD\_CONSTANT\_INTERPOLATION, FIELD\_DEPENDENT\_TYPE, FIELD\_ELEMENT\_BASED\_INTERPOLATION, FIELD\_ELEMENT\_DOF\_TYPE, FIELD\_FIBRE\_TYPE, FIELD\_GAUSS\_POINT\_BASED\_INTERPOLATION, FIELD\_GENERAL\_TYPE, FIELD\_GEOMETRIC\_TYPE, FIELD\_GRID\_POINT\_BASED\_INTERPOLATION, FIELD\_HARMONIC\_MEAN\_SCALING, FIELD\_INDEPENDENT\_TYPE, FIELD\_INTERPOLATE\_XI(), FIELD\_INTERPOLATED\_POINT\_FINALISE(), FIELD\_INTERPOLATED\_POINT\_INITIALISE(), FIELD\_INTERPOLATION\_PARAMETERS\_FINALISE(), FIELD\_INTERPOLATION\_PARAMETERS\_INITIALISE(), FIELD\_MATERIAL\_TYPE, FIELD\_NO\_SCALING, FIELD\_NODE\_BASED\_INTERPOLATION, FIELD\_NODE\_DOF\_TYPE, FIELD\_NUMBER\_OF\_SET\_TYPES, FIELD\_SCALAR\_DIMENSION\_TYPE, FIELD\_STANDARD\_VARIABLE\_TYPE, FIELD\_UNIT\_SCALING, FIELD\_VALUES\_SET\_TYPE, FIELD\_VECTOR\_DIMENSION\_TYPE, LISTS::LIST\_CREATE\_FINISH(), LISTS::LIST\_CREATE\_START(), LISTS::LIST\_DATA\_TYPE\_SET(), LISTS::LIST\_DESTROY(), LISTS::LIST\_INITIAL\_SIZE\_SET(), LISTS::LIST\_INTG\_TYPE, LISTS::LIST\_REMOVE\_DUPLICATES(), MATRIX\_VECTOR::MATRIX\_VECTOR\_DP\_TYPE, NODE\_ROUTINES::NODE\_INITIAL\_POSITION\_SET(), NODE\_ROUTINES::NODES\_CREATE\_FINISH(), and KINDS::PTR.

Here is the call graph for this function:

```
6.20.2.27 subroutine FIELD_ROUTINES::FIELD_VARIABLE_COMPONENT_FINALISE (TYPE(FIELD_VARIABLE_COMPONENT_TYPE)
 &FIELD_VARIABLE_COMPONENT, INTEGER(INTG),intent(out) ERR,
 &TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Finalises a field variable component and deallocates all memory.

#### Parameters:

**FIELD\_VARIABLE\_COMPONENT** The field variable component to finalise  
**ERR** The error code  
**ERROR** The error string

Definition at line 596 of file field\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by FIELD\_VARIABLE\_COMPONENT\_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.20.2.28 subroutine FIELD\_ROUTINES::FIELD\_VARIABLE\_COMPONENT\_INITIALISE**  
 (**TYPE(FIELD\_TYPE)**,pointer **FIELD**, **INTEGER(INTG)**,**intent(in)**  
**VARIABLE\_NUMBER**, **INTEGER(INTG)**,**intent(in)** **COMPONENT\_NUMBER**,  
**INTEGER(INTG)**,**intent(out)** **ERR**, **TYPE(VARYING\_STRING)**,**intent(out)** **ERROR**,  
 \*) [private]

Initialises a field variable component.

**Parameters:**

**FIELD** A pointer to the field containing the field variable component to initialise  
**VARIABLE\_NUMBER** The field variable number of the field variable component  
**COMPONENT\_NUMBER** The field component number of the field variable component  
**ERR** The error code  
**ERROR** The error string

Definition at line 620 of file field\_routines.f90.

References **BASE\_ROUTINES::ENTERS()**, **BASE\_ROUTINES::ERRORS()**, **BASE\_ROUTINES::EXITS()**, **FIELD\_CONSTANT\_INTERPOLATION**, **FIELD\_ELEMENT\_BASED\_INTERPOLATION**, **FIELD\_GAUSS\_POINT\_BASED\_INTERPOLATION**, **FIELD\_GRID\_POINT\_BASED\_INTERPOLATION**, **FIELD\_NODE\_BASED\_INTERPOLATION**, **FIELD\_VARIABLE\_COMPONENT\_FINALISE()**, and **KINDS::PTR**.

Here is the call graph for this function:

**6.20.2.29 subroutine FIELD\_ROUTINES::FIELD\_VARIABLES\_FINALISE**  
 (**TYPE(FIELD\_TYPE)**,pointer **FIELD**, **INTEGER(INTG)**,**intent(out)** **ERR**,  
**TYPE(VARYING\_STRING)**,**intent(out)** **ERROR**, \*) [private]

Finalises the field variables for a field and deallocates all memory.

**Parameters:**

**FIELD** A pointer to the field to finalise the variables for  
**ERR** The error code  
**ERROR** The error string

Definition at line 5897 of file field\_routines.f90.

References **BASE\_ROUTINES::ENTERS()**, **BASE\_ROUTINES::ERRORS()**, and **BASE\_ROUTINES::EXITS()**.

Referenced by **FIELD\_DESTROY()**.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.20.2.30 subroutine FIELD\_ROUTINES::FIELD\_VARIABLES\_INITIALISE**  
 (**TYPE(FIELD\_TYPE)**,pointer **FIELD**, **INTEGER(INTG)**,**intent(out)** **ERR**,  
**TYPE(VARYING\_STRING)**,**intent(out)** **ERROR**, \*) [private]

Initialises the field variables.

**Parameters:**

***FIELD*** A pointer to the field to initialise the variables for

***ERR*** The error code

***ERROR*** The error string

Definition at line 5932 of file field\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `FIELD_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.20.2.31 subroutine `FIELD_ROUTINES::FIELDS_FINALISE` (`TYPE(REGION_TYPE),pointer REGION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, */`)

Finalises the fields for the given region and deallocates all memory.

**Parameters:**

***REGION*** A pointer to the region to finalise the fields for

***ERR*** The error code

***ERROR*** The error string

Definition at line 5969 of file field\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `FIELD_DESTROY()`, and `KINDS::PTR`.

Referenced by `REGION_ROUTINES::REGION_DESTROY()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.20.2.32 subroutine `FIELD_ROUTINES::FIELDS_INITIALISE` (`TYPE(REGION_TYPE),pointer REGION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, */`)

Initialises the fields for the given region.

**Parameters:**

***REGION*** A pointer to the region to initialise the fields for

***ERR*** The error code

***ERROR*** The error string

Definition at line 6004 of file field\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by REGION\_ROUTINES::REGION\_CREATE\_START(), and REGION\_-ROUTINES::REGION\_SUB\_REGION\_CREATE\_START().

Here is the call graph for this function:

Here is the caller graph for this function:

## 6.21 FINITE\_ELASTICITY\_ROUTINES Namespace Reference

This module handles all finite elasticity routines.

### Functions

- subroutine [FINITE\\_ELASTICITYFINITE\\_ELEMENT\\_JACOBIAN\\_EVALUATE](#) (EQUATIONS\_SET, ELEMENT\_NUMBER, ERR, ERROR,\*)
 

*Evaluates the Jacobian for a finite elasticity finite element equations set.*
- subroutine [FINITE\\_ELASTICITYFINITE\\_ELEMENT\\_RESIDUAL\\_EVALUATE](#) (EQUATIONS\_SET, ELEMENT\_NUMBER, ERR, ERROR,\*)
 

*Evaluates the residual and RHS vectors for a finite elasticity finite element equations set.*
- subroutine [FINITE\\_ELASTICITYGAUSS\\_DEFORMATION\\_GRADEINT\\_TENSOR](#) (INTERPOLATED\_POINT\_DEF, INT RPOLATED\_POINT\_FIBRE,&INTERPOLATED\_POINT\_UNDEF, ELEMENT\_NUMBER, GAUSS\_PT\_NUMBER, DZDNU, Jxxi, Jznu, ERR, ERROR,\*)
 

*Evaluates the deformation gradient tensor at a given Gauss point.*
- subroutine [FINITE\\_ELASTICITYGAUSS\\_DXDNU](#) (INTERPOLATED\_POINT, DXDNU, DXDXI, ERR, ERROR,\*)
 

*Evaluates dx/dnu(uniformed-material cs) tensor at a given Gauss point.*
- subroutine [FINITE\\_ELASTICITYGAUSS\\_CAUCHY\\_TENSOR](#) (FIELD, INTERPOLATED\_POINT, CAUCHY\_TENSOR, DZDNU, ERR, ERROR,\*)
 

*Evaluates the Cauchy stress tensor at a given Gauss point.*
- subroutine [FINITE\\_ELASTICITYGAUSS\\_DFDZ](#) (INTERPOLATED\_POINT, DFDZ, XI, ERR, ERROR,\*)
 

*Evaluates df/dz (derivative of interpolation function wrt deformed coord) matrix at a given Gauss point.*
- subroutine [FINITE\\_ELASTICITYEQUATIONS\\_SET\\_SETUP](#) (EQUATIONS\_SET, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)
 

*Sets up the finite elasticity equation type of an elasticity equations set class.*
- subroutine [FINITE\\_ELASTICITYEQUATIONS\\_SET\\_SUBTYPE\\_SET](#) (EQUATIONS\_SET, EQUATIONS\_SET\_SUBTYPE, ERR, ERROR,\*)
 

*Sets/changes the equation subtype for a finite elasticity equation type of an elasticity equations set class.*
- subroutine [FINITE\\_ELASTICITYPROBLEM\\_SETUP](#) (PROBLEM, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)
 

*Sets up the finite elasticity problem.*
- subroutine [FINITE\\_ELASTICITYPROBLEM\\_SUBTYPE\\_SET](#) (PROBLEM, PROBLEM\_SUBTYPE, ERR, ERROR,\*)
 

*Sets/changes the problem subtype for a finite elasticity type .*

### 6.21.1 Detailed Description

This module handles all finite elasticity routines.

### 6.21.2 Function Documentation

**6.21.2.1 subroutine FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITY\_-  
EQUATIONS\_SET\_SETUP (TYPE(EQUATIONS\_SET\_TYPE),pointer  
EQUATIONS\_SET, INTEGER(INTG),intent(in) SETUP\_TYPE,  
INTEGER(INTG),intent(in) ACTION\_TYPE, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets up the finite elasticity equation type of an elasticity equations set class.

**Parameters:**

**EQUATIONS\_SET** A pointer to the equations set to setup a Laplace equation on.  
**SETUP\_TYPE** The setup type  
**ACTION\_TYPE** The action type  
**ERR** The error code  
**ERROR** The error string

Definition at line 580 of file finite\_elasticity\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_CREATE_FINISH()`, `EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_CREATE_START()`, `EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_LINEAR_MATRICES_NUMBER_SET()`, `EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_RHS_VARIABLE_TYPE_SET()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_BEM SOLUTION_METHOD`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FD SOLUTION_METHOD`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FEM SOLUTION_METHOD`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FV SOLUTION_METHOD`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_GFEM SOLUTION_METHOD`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NO_SUBTYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NONLINEAR`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_ANALYTIC_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_DEPENDENT_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_EQUIVATIONS_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FINISH_ACTION`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FIXED_CONDITIONS_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_GEOMETRY_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_INITIAL_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_MATERIALS_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_SOURCE_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_START_ACTION`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_STATIC`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_ROUTINES::FIELD_BOUNDARY_CONDITIONS_SET_TYPE`, `FIELD_ROUTINES::FIELD_CREATE_FINISH()`, `FIELD_ROUTINES::FIELD_DEPENDENT_TYPE`, `FIELD_ROUTINES::FIELD_GEOMETRIC_TYPE`, `FIELD_ROUTINES::FIELD_NODE_BASED_INTERPOLATION`, `FIELD_ROUTINES::FIELD_NORMAL_VARIABLE_TYPE`, and `FIELD_ROUTINES::FIELD_STANDARD_VARIABLE_TYPE`.

Referenced by ELASTICITY\_ROUTINES::ELASTICITY\_EQUATIONS\_SET\_SETUP(), and FINITE\_ELASTICITY\_EQUATIONS\_SET\_SUBTYPE\_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.21.2.2 subroutine FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITY\_EQUATIONS\_SET\_SUBTYPE\_SET (TYPE(EQUATIONS\_SET\_TYPE),pointer EQUATIONS\_SET, INTEGER(INTG),intent(in) EQUATIONS\_SET\_SUBTYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets/changes the equation subtype for a finite elasticity equation type of an elasticity equations set class.

**Parameters:**

**EQUATIONS\_SET** A pointer to the equations set to set the equation subtype for

**EQUATIONS\_SET\_SUBTYPE** The equation subtype to set

**ERR** The error code

**ERROR** The error string

Definition at line 820 of file finite\_elasticity\_routines.f90.

References BASE\_ROUTINES::ENTERS(), EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_ELASTICITY\_CLASS, EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SETFINITE\_ELASTICITY\_TYPE, EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_NO\_SUBTYPE, EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_SETUP\_INITIAL\_TYPE, EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_SETUP\_START\_ACTION, BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and FINITE\_ELASTICITY\_EQUATIONS\_SET\_SETUP().

Referenced by ELASTICITY\_ROUTINES::EL().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.21.2.3 subroutine FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITYFINITE\_ELEMENT\_JACOBIAN\_EVALUATE (TYPE(EQUATIONS\_SET\_TYPE),pointer EQUATIONS\_SET, INTEGER(INTG),intent(in) ELEMENT\_NUMBER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Evaluates the Jacobian for a finite elasticity finite element equations set.

**Parameters:**

**EQUATIONS\_SET** A pointer to the equations set to perform the finite element calculations on

**ELEMENT\_NUMBER** The element number to evaluate

**ERR** The error code

**ERROR** The error string

Definition at line 90 of file finite\_elasticity\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by ELASTICITY\_ROUTINES::ELASTICITYFINITEELEMENTJACOBIAN\_-  
EVALUATE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.21.2.4 subroutine FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITYFINITE\_-  
ELEMENT\_RESIDUAL\_EVALUATE (TYPE(EQUATIONS\_SET\_TYPE),pointer  
EQUATIONS\_SET, INTEGER(INTG),intent(in) ELEMENT\_NUMBER,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Evaluates the residual and RHS vectors for a finite elasticity finite element equations set.

**Parameters:**

**EQUATIONS\_SET** A pointer to the equations set to perform the finite element calculations on

**ELEMENT\_NUMBER** The element number to calculate

**ERR** The error code

**ERROR** The error string

Definition at line 125 of file finite\_elasticity\_routines.f90.

References KINDS::DP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(),  
BASE\_ROUTINES::EXITS(), FIELD\_ROUTINES::FIELD\_CREATE\_FINISH(), FIELD\_-  
ROUTINES::FIELD\_ELEMENT\_BASED\_INTERPOLATION, FIELD\_ROUTINES::FIELD\_-  
GENERAL\_TYPE, FIELD\_ROUTINES::FIELD\_INTERPOLATE\_XI(), FIELD\_ROUTINES::FIELD\_-  
INTERPOLATED\_POINT\_INITIALISE(), FIELD\_ROUTINES::FIELD\_INTERPOLATION\_-  
PARAMETERS\_ELEMENT\_GET(), FIELD\_ROUTINES::FIELD\_INTERPOLATION\_-  
PARAMETERS\_INITIALISE(), FINITE\_ELASTICITY\_GAUSS\_CAUCHY\_TENSOR(), FINITE\_-  
ELASTICITY\_GAUSS\_DEFORMATION\_GRADEINT\_TENSOR(), FINITE\_ELASTICITY\_GAUSS\_-  
DFDZ(), and KINDS::PTR.

Referenced by ELASTICITY\_ROUTINES::ELASTICITYFINITEELEMENTRESIDUAL\_-  
EVALUATE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.21.2.5 subroutine FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITY\_-  
GAUSS\_CAUCHY\_TENSOR (TYPE(FIELD\_TYPE),pointer FIELD,  
TYPE(FIELD\_INTERPOLATED\_POINT\_TYPE),pointer INTERPOLATED\_-  
POINT, REAL(DP),dimension(:, :, intent(out) CAUCHY\_TENSOR,  
REAL(DP),dimension(:, :, intent(in) DZDNU, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Evaluates the Cauchy stress tensor at a given Gauss point.

**Parameters:**

**ERR** The error code

**ERROR** The error string

Definition at line 443 of file finite\_elasticity\_routines.f90.

References KINDS::DP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and KINDS::PTR.

Referenced by FINITE\_ELASTICITYFINITE\_ELEMENT\_RESIDUAL\_EVALUATE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.21.2.6 subroutine FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITY\_GAUSS\_DEFORMATION\_GRADEINT\_TENSOR**  
`(TYPE(FIELD_INTERPOLATED_POINT_TYPE),pointer INTERPOLATED_POINT_DEF, INT RPOLATED_POINT_FIBRE, &,pointer INTERPOLATED_POINT_UNDEF, INTEGER(INTG) ELEMENT_NUMBER, INTEGER(INTG) GAUSS_PT_NUMBER, REAL(DP),dimension(3,3) DZDNU, REAL(DP) Jxxi, REAL(DP) Jznu, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Evaluates the deformation gradient tensor at a given Gauss point.

#### Parameters:

**ERR** The error code

**ERROR** The error string

Definition at line 307 of file finite\_elasticity\_routines.f90.

References KINDS::DP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and FINITE\_ELASTICITY\_GAUSS\_DXDNU().

Referenced by FINITE\_ELASTICITYFINITE\_ELEMENT\_RESIDUAL\_EVALUATE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.21.2.7 subroutine FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITY\_GAUSS\_DFDZ**  
`(TYPE(FIELD_INTERPOLATED_POINT_TYPE),pointer INTERPOLATED_POINT, REAL(DP),dimension(:,,:),allocatable DFDZ, REAL(DP),dimension(:,),allocatable XI, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Evaluates df/dz (derivative of interpolation function wrt deformed coord) matrix at a given Gauss point.

#### Parameters:

**ERR** The error code

**ERROR** The error string

Definition at line 499 of file finite\_elasticity\_routines.f90.

References KINDS::DP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and KINDS::PTR.

Referenced by FINITE\_ELASTICITYFINITE\_ELEMENT\_RESIDUAL\_EVALUATE().

Here is the call graph for this function:

Here is the caller graph for this function:

```
6.21.2.8 subroutine FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_-
GAUSS_DXNU (TYPE(FIELD_INTERPOLATED_POINT_TYPE),pointer
INTERPOLATED_POINT, REAL(DP),dimension(:,:) DXNU,
REAL(DP),dimension(:,:) DXDXI, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Evaluates dx/dnu(undeformed-material cs) tensor at a given Gauss point.

**Parameters:**

**ERR** The error code

**ERROR** The error string

Definition at line 355 of file finite\_elasticity\_routines.f90.

References KINDS::\_DP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by FINITE\_ELASTICITY\_GAUSS\_DEFORMATION\_GRADEINT\_TENSOR().

Here is the call graph for this function:

Here is the caller graph for this function:

```
6.21.2.9 subroutine FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_-
PROBLEM_SETUP (TYPE(PROBLEM_TYPE),pointer PROBLEM,
INTEGER(INTG),intent(in) SETUP_TYPE, INTEGER(INTG),intent(in) ACTION_-
TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)
ERROR, *)
```

Sets up the finite elasticity problem.

**Parameters:**

**PROBLEM** A pointer to the solutions set to setup a Laplace equation on.

**SETUP\_TYPE** The setup type

**ACTION\_TYPE** The action type

**ERR** The error code

**ERROR** The error string

Definition at line 861 of file finite\_elasticity\_routines.f90.

References BASE\_ROUTINES::ENTERS(), EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_ELASTICITY\_CLASS, EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SETFINITE\_ELASTICITY\_TYPE, EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_NO\_SUBTYPE, BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), PROBLEM\_CONSTANTS::PROBLEM\_NO\_SUBTYPE, PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_CONTROL\_TYPE, PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_DO\_ACTION, PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_FINISH\_ACTION, PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_INITIAL\_TYPE, PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_SOLUTION\_TYPE, PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_SOLVER\_TYPE, PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_START\_ACTION,

PROBLEM\_CONSTANTS::PROBLEM SOLUTION\_NONLINEAR, KINDS::PTR, SOLVER\_ROUTINES::SOLVER\_CREATE\_FINISH(), SOLVER\_ROUTINES::SOLVER\_CREATE\_START(), SOLVER\_ROUTINES::SOLVER\_LIBRARY\_SET(), SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_TYPE, and SOLVER\_ROUTINES::SOLVER\_PETSC\_LIBRARY.

Referenced by ELASTICITY\_ROUTINES::ELASTICITY\_PROBLEM\_SETUP(), and FINITE\_ELASTICITY\_PROBLEM\_SUBTYPE\_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.21.2.10 subroutine FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITY\_PROBLEM\_SUBTYPE\_SET (TYPE(PROBLEM\_TYPE),pointer *PROBLEM*, INTEGER(INTG),intent(in) *PROBLEM\_SUBTYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Sets/changes the problem subtype for a finite elasticity type .

#### Parameters:

***PROBLEM*** A pointer to the problem to set the problem subtype for

***PROBLEM\_SUBTYPE*** The problem subtype to set

***ERR*** The error code

***ERROR*** The error string

Definition at line 1004 of file finite\_elasticity\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), FINITE\_ELASTICITY\_PROBLEM\_SETUP(), PROBLEM\_CONSTANTS::PROBLEM\_ELASTICITY\_CLASS, PROBLEM\_CONSTANTS::PROBLEM\_FINITE\_ELASTICITY\_TYPE, PROBLEM\_CONSTANTS::PROBLEM\_NO\_SUBTYPE, PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_INITIAL\_TYPE, and PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_START\_ACTION.

Referenced by ELASTICITY\_ROUTINES::ELASTICITY\_PROBLEM\_CLASS\_TYPE\_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

## 6.22 FINITE\_ELEMENT\_ROUTINES Namespace Reference

This module contains all finite element base routines.

### Functions

- subroutine `FEM_ELEMENT_MATRICES_FINALISE` (`ELEMENT_MATRICES`, `ERR`, `ERROR,*`)
- subroutine `FEM_ELEMENT_MATRICES_INITIALISE` (`PROBLEM`, `ELEMENT_MATRICES`, `ERR`, `ERROR,*`)

#### 6.22.1 Detailed Description

This module contains all finite element base routines.

#### 6.22.2 Function Documentation

**6.22.2.1 subroutine `FINITE_ELEMENT_ROUTINES::FEM_ELEMENT_-  
MATRICES_FINALISE` (`TYPE(FEM_ELEMENT_MATRICES_TYPE),pointer  
ELEMENT_MATRICES`, `INTEGER(INTG),intent(out) ERR`,  
`TYPE(VARYING_STRING),intent(out) ERROR, *`)**

Definition at line 72 of file `finite_element_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_-ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.22.2.2 subroutine `FINITE_ELEMENT_ROUTINES::FEM_ELEMENT_-  
MATRICES_INITIALISE` (`TYPE(PROBLEM_TYPE),pointer PROBLEM`,  
`TYPE(FEM_ELEMENT_MATRICES_TYPE),pointer ELEMENT_MATRICES`,  
`INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, *`)  
[private]**

Definition at line 104 of file `finite_element_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_-ROUTINES::EXITS()`.

Here is the call graph for this function:

## 6.23 FLUID\_MECHANICS\_ROUTINES Namespace Reference

This module handles all fluid mechanics class routines.

### 6.23.1 Detailed Description

This module handles all fluid mechanics class routines.

## 6.24 GENERATED\_MESH\_ROUTINES Namespace Reference

This module handles all generated mesh routines.

### Classes

- interface [GENERATED\\_MESH\\_BASIS\\_SET](#)
- interface [GENERATED\\_MESH\\_DESTROY](#)
- interface [GENERATED\\_MESH\\_EXTENT\\_SET](#)
- interface [GENERATED\\_MESH\\_NUMBER\\_OF\\_ELEMENTS\\_SET](#)
- interface [GENERATED\\_MESH\\_ORIGIN\\_SET](#)
- interface [GENERATED\\_MESH\\_TYPE\\_SET](#)

### Functions

- TYPE([BASIS\\_TYPE](#)) [GENERATED\\_MESH\\_BASIS\\_GET](#) (GENERATED\_MESH, ERR, ERROR)
 

*Gets the basis of a generated mesh.*
- subroutine [GENERATED\\_MESH\\_BASIS\\_SET\\_NUMBER](#) (USER\_NUMBER, BASIS, ERR, ERROR,\*)
 

*Set the basis of the generated mesh.*
- subroutine [GENERATED\\_MESH\\_BASIS\\_SET\\_PTR](#) (GENERATED\_MESH, BASIS, ERR, ERROR,\*)
 

*Sets/changes the basis of a generated mesh.*
- subroutine [GENERATED\\_MESH\\_CREATE\\_FINISH](#) (GENERATED\_MESH, MESH, MESH\_USER\_NUMBER, ERR, ERROR,\*)
 

*Finishes the creation of a generated mesh.*
- subroutine [GENERATED\\_MESH\\_CREATE\\_START](#) (USER\_NUMBER, REGION, GENERATED\_MESH, ERR, ERROR,\*)
 

*Starts the creation of a generated mesh.*
- subroutine [GENERATED\\_MESH\\_DESTROY\\_NUMBER](#) (USER\_NUMBER, ERR, ERROR,\*)
 

*Destroys a generated mesh.*
- subroutine [GENERATED\\_MESH\\_DESTROY\\_PTR](#) (GENERATED\_MESH, ERR, ERROR,\*)
 

*Destroys a generated mesh.*
- REAL(DP), pointer [GENERATED\\_MESH\\_EXTENT\\_GET](#) (GENERATED\_MESH, ERR, ERROR)
 

*Gets the extent of a generated mesh.*
- subroutine [GENERATED\\_MESH\\_EXTENT\\_SET\\_NUMBER](#) (USER\_NUMBER, MAX\_EXTENT, ERR, ERROR,\*)
 

*Sets/changes the max extent of a generated mesh.*

- subroutine [GENERATED\\_MESH\\_EXTENT\\_SET\\_PTR](#) (GENERATED\_MESH, MAX\_EXTENT, ERR, ERROR,\*)
 

*Sets/changes the extent of a generated mesh.*
- subroutine [GENERATED\\_MESH\\_FINALISE](#) (GENERATED\_MESH, ERR, ERROR,\*)
 

*Finalises a generated mesh and deallocates all memory.*
- subroutine [GENERATED\\_MESH\\_INITIALISE](#) (GENERATED\_MESH, ERR, ERROR,\*)
 

*Initialises a generated mesh.*
- INTEGER(INTG), pointer [GENERATED\\_MESH\\_NUMBER\\_OF\\_ELEMENTS\\_GET](#) (GENERATED\_MESH, ERR, ERROR)
 

*Gets the extent of a generated mesh.*
- subroutine [GENERATED\\_MESH\\_NUMBER\\_OF\\_ELEMENTS\\_SET\\_NUMBER](#) (USER\_NUMBER, NUMBER\_OF\_ELEMENTS\_XI, ERR, ERROR,\*)
 

*Sets/changes the extent of a generated mesh.*
- subroutine [GENERATED\\_MESH\\_NUMBER\\_OF\\_ELEMENTS\\_SET\\_PTR](#) (GENERATED\_MESH, NUMBER\_OF\_ELEMENTS\_XI, ERR, ERROR,\*)
 

*Sets/changes the extent of a generated mesh.*
- REAL(DP), pointer [GENERATED\\_MESH\\_ORIGIN\\_GET](#) (GENERATED\_MESH, ERR, ERROR)
 

*Get the origin of a generated mesh.*
- subroutine [GENERATED\\_MESH\\_ORIGIN\\_SET\\_NUMBER](#) (USER\_NUMBER, ORIGIN, ERR, ERROR,\*)
 

*Sets/changes the origin of a generated mesh.*
- subroutine [GENERATED\\_MESH\\_ORIGIN\\_SET\\_PTR](#) (GENERATED\_MESH, ORIGIN, ERR, ERROR,\*)
 

*Sets/changes the origin of a generated mesh.*
- subroutine [GENERATED\\_MESH\\_REGULAR\\_CREATE\\_FINISH](#) (GENERATED\_MESH, MESH\_USER\_NUMBER, ERR, ERROR,\*)
 

*Start to create the regular generated mesh type.*
- subroutine [GENERATED\\_MESH\\_REGULAR\\_FINALISE](#) (REGULAR\_MESH, ERR, ERROR,\*)
 

*Finalise the regular mesh type.*
- subroutine [GENERATED\\_MESH\\_REGULAR\\_INITIALISE](#) (GENERATED\_MESH, ERR, ERROR,\*)
 

*Initialise the regular generated mesh type.*
- INTEGER(INTG) [GENERATED\\_MESH\\_TYPE\\_GET](#) (GENERATED\_MESH, ERR, ERROR)
 

*Gets the type of a generated mesh.*
- subroutine [GENERATED\\_MESH\\_TYPE\\_SET\\_NUMBER](#) (USER\_NUMBER, GENERATED\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the type of a generated mesh.*

- subroutine `GENERATED_MESH_TYPE_SET_PTR` (`GENERATED_MESH`, `GENERATED_MESH_TYPE`, `ERR`, `ERROR,*`)  
*Sets/changes the type of a generated mesh.*
- subroutine `GENERATED_MESH_USER_NUMBER_FIND` (`USER_NUMBER`, `GENERATED_MESH`, `ERR`, `ERROR,*`)  
*Finds and returns in generated mesh a pointer to that identified by `USER_NUMBER`. If no generated mesh with that number exists left nullified.*
- subroutine `GENERATED_MESHES_FINALISE` (`ERR`, `ERROR,*`)  
*Finalises all generated meshes and deallocates all memory.*
- subroutine `GENERATED_MESHES_INITIALISE` (`ERR`, `ERROR,*`)  
*Initialises all generated meshes.*

## Variables

- INTEGER(INTG), parameter `GENERATED_MESH_REGULAR_MESH_TYPE` = 1  
*A regular generated mesh.*
- INTEGER(INTG), parameter `GENERATED_MESH_POLAR_MESH_TYPE` = 2  
*A polar generated mesh.*
- INTEGER(INTG), parameter `GENERATED_MESH_FRACTAL_TREE_MESH_TYPE` = 3  
*A fractal tree generated mesh.*
- TYPE(`GENERATED_MESHES_TYPE`), target `GENERATED_MESHES`

### 6.24.1 Detailed Description

This module handles all generated mesh routines.

### 6.24.2 Function Documentation

#### 6.24.2.1 `TYPE(BASIS_TYPE) GENERATED_MESH_ROUTINES::GENERATED_MESH_BASIS_GET (TYPE(GENERATED_MESH_TYPE),pointer GENERATED_MESH, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)` [private]

Gets the basis of a generated mesh.

#### Parameters:

**`GENERATED_MESH`** A pointer to the generated mesh to get the basis of

**`ERR`** The error code

**`ERROR`** The error string

Definition at line 127 of file generated\_mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `GENERATED_MESH_REGULAR_MESH_TYPE`.

Here is the call graph for this function:

**6.24.2.2 subroutine `GENERATED_MESH_ROUTINES::GENERATED_MESH_BASIS_SET_NUMBER` (`INTEGER(INTG),intent(in) USER_NUMBER`, `TYPE(BASIS_TYPE),pointer BASIS`, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR`, \*) [private]**

Set the basis of the generated mesh.

Definition at line 165 of file generated\_mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `GENERATED_MESH_BASIS_SET_PTR()`, and `GENERATED_MESH_USER_NUMBER_FIND()`.

Here is the call graph for this function:

**6.24.2.3 subroutine `GENERATED_MESH_ROUTINES::GENERATED_MESH_BASIS_SET_PTR` (`TYPE(GENERATED_MESH_TYPE),pointer GENERATED_MESH`, `TYPE(BASIS_TYPE),pointer BASIS`, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR`, \*) [private]**

Sets/changes the basis of a generated mesh.

#### Parameters:

**`GENERATED_MESH`** A pointer to the generated mesh to set the basis of

**`BASIS`** The basis of mesh to generate

#### See also:

`GENERATED_MESH_ROUTINES_GeneratedMeshBasis`, [GENERATED\\_MESH\\_ROUTINES](#)

**`ERR`** The error code

**`ERROR`** The error string

Definition at line 196 of file generated\_mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `GENERATED_MESH_REGULAR_MESH_TYPE`.

Referenced by `GENERATED_MESH_BASIS_SET_NUMBER()`.

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.24.2.4 subroutine GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_CREATE\_-  
FINISH (TYPE(GENERATED\_MESH\_TYPE),pointer *GENERATED\_MESH*,  
TYPE(MESH\_TYPE),pointer *MESH*, INTEGER(INTG),intent(in) *MESH\_USER\_-  
NUMBER*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out)  
*ERROR*, \*)**

Finishes the creation of a generated mesh.

**Parameters:**

***GENERATED\_MESH*** A pointer to the generated mesh to finish the creation of  
***MESH*** A pointer to the generated mesh to finish the creation of  
***MESH\_USER\_NUMBER*** The mesh's user number  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 243 of file generated\_mesh\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    **BASE\_-  
ROUTINES::EXITS()**, **GENERATED\_MESH\_REGULAR\_CREATE\_FINISH()**, and **GENERATED\_-  
MESH\_REGULAR\_MESH\_TYPE**.

Here is the call graph for this function:

**6.24.2.5 subroutine GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_CREATE\_-  
START (INTEGER(INTG),intent(in) *USER\_NUMBER*, TYPE(REGION\_TYPE),pointer  
*REGION*, TYPE(GENERATED\_MESH\_TYPE),pointer *GENERATED\_MESH*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Starts the creation of a generated mesh.

**Parameters:**

***USER\_NUMBER*** The user number of the generated mesh to create  
***REGION*** A pointer to the region to create the generated mesh on  
***GENERATED\_MESH*** On exit, a pointer to the created generated mesh. Must not be associated on  
entry.  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 291 of file generated\_mesh\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    **BASE\_-  
ROUTINES::EXITS()**,    **GENERATED\_MESH\_INITIALISE()**,    **GENERATED\_MESHES**, and  
**KINDS::PTR**.

Here is the call graph for this function:

**6.24.2.6 subroutine GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_DESTROY\_-  
NUMBER (INTEGER(INTG),intent(in) *USER\_NUMBER*, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Destroys a generated mesh.

**Parameters:**

**USER\_NUMBER** The user number of generated mesh to destroy

**ERR** The error code

**ERROR** The error string

Definition at line 353 of file generated\_mesh\_routines.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(), GENERATED\_MESHES, and KINDS::PTR.

Here is the call graph for this function:

**6.24.2.7 subroutine GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_DESTROY\_-PTR (TYPE(GENERATED\_MESH\_TYPE),pointer GENERATED\_MESH,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)  
[private]**

Destroys a generated mesh.

**Parameters:**

**GENERATED\_MESH** A pointer to the generated mesh to destroy

**ERR** The error code

**ERROR** The error string

Definition at line 400 of file generated\_mesh\_routines.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(),     GENERATED\_MESH\_FINALISE(),     GENERATED\_MESHES, and KINDS::PTR.

Here is the call graph for this function:

**6.24.2.8 REAL(DP),pointer GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_-EXTENT\_GET (TYPE(GENERATED\_MESH\_TYPE),pointer GENERATED\_MESH,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR)**

Gets the extent of a generated mesh.

**Parameters:**

**GENERATED\_MESH** A pointer to the generated mesh to get the type of

**ERR** The error code

**ERROR** The error string

Definition at line 456 of file generated\_mesh\_routines.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(), and GENERATED\_MESH\_REGULAR\_MESH\_TYPE.

Here is the call graph for this function:

---

**6.24.2.9 subroutine GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_-  
EXTENT\_SET\_NUMBER (INTEGER(INTG),intent(in) *USER\_NUMBER*,  
REAL(DP),dimension(:),intent(in) *MAX\_EXTENT*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Sets/changes the max extent of a generated mesh.

Definition at line 493 of file generated\_mesh\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(),    GENERATED\_MESH\_EXTENT\_SET\_PTR(), and    GENERATED\_MESH\_-USER\_NUMBER\_FIND().

Here is the call graph for this function:

**6.24.2.10 subroutine GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_EXTENT\_-  
SET\_PTR (TYPE(GENERATED\_MESH\_TYPE),pointer *GENERATED\_MESH*,  
REAL(DP),dimension(:),intent(in) *MAX\_EXTENT*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Sets/changes the extent of a generated mesh.

#### Parameters:

***GENERATED\_MESH*** A pointer to the generated mesh to set the type of  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 524 of file generated\_mesh\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), and GENERATED\_MESH\_REGULAR\_MESH\_TYPE.

Referenced by GENERATED\_MESH\_EXTENT\_SET\_NUMBER().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.24.2.11 subroutine GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_FINALISE  
(TYPE(GENERATED\_MESH\_TYPE),pointer *GENERATED\_MESH*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
\*) [private]**

Finalises a generated mesh and deallocates all memory.

#### Parameters:

***GENERATED\_MESH*** A pointer to the generated mesh to finalise.  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 567 of file generated\_mesh\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), and GENERATED\_MESH\_REGULAR\_FINALISE().

Referenced by GENERATED\_MESH\_DESTROY\_PTR().

Here is the call graph for this function:

Here is the caller graph for this function:

```
6.24.2.12 subroutine GENERATED_MESH_ROUTINES::GENERATED_MESH_INITIALISE

 (TYPE(GENERATED_MESH_TYPE),pointer GENERATED_MESH,

 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,

 *) [private]
```

Initialises a generated mesh.

**Parameters:**

**GENERATED\_MESH** A pointer to the generated mesh to initialise

**ERR** The error code

**ERROR** The error string

Definition at line 596 of file generated\_mesh\_routines.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(), and GENERATED\_MESH\_REGULAR\_INITIALISE().

Referenced by GENERATED\_MESH\_CREATE\_START().

Here is the call graph for this function:

Here is the caller graph for this function:

```
6.24.2.13 INTEGER(INTG),pointer GENERATED_MESH_ROUTINES::GENERATED_-

 MESH_NUMBER_OF_ELEMENTS_GET (TYPE(GENERATED_MESH_-

 TYPE),pointer GENERATED_MESH, INTEGER(INTG),intent(out) ERR,

 TYPE(VARYING_STRING),intent(out) ERROR)
```

Gets the extent of a generated mesh.

**Parameters:**

**GENERATED\_MESH** A pointer to the generated mesh to set the type of

**ERR** The error code

**ERROR** The error string

Definition at line 631 of file generated\_mesh\_routines.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(), and GENERATED\_MESH\_REGULAR\_MESH\_TYPE.

Here is the call graph for this function:

```
6.24.2.14 subroutine GENERATED_MESH_ROUTINES::GENERATED_MESH_NUMBER_-

 OF_ELEMENTS_SET_NUMBER (INTEGER(INTG),intent(in) USER_NUMBER,

 INTEGER(INTG),dimension(:),intent(in) NUMBER_OF_ELEMENTS_XI,

 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,

 *) [private]
```

Definition at line 667 of file generated\_mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `GENERATED_MESH_NUMBER_OF_ELEMENTS_SET_PTR()`,    and `GENERATED_MESH_USER_NUMBER_FIND()`.

Here is the call graph for this function:

**6.24.2.15 subroutine `GENERATED_MESH_ROUTINES::GENERATED_MESH_NUMBER_OF_ELEMENTS_SET_PTR`** (`TYPE(GENERATED_MESH_TYPE),pointer GENERATED_MESH, INTEGER(INTG),dimension(:),intent(in) NUMBER_OF_ELEMENTS_XI, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, */ [private]`)

Sets/changes the extent of a generated mesh.

**Parameters:**

**`GENERATED_MESH`** A pointer to the generated mesh to set the type of  
**`ERR`** The error code  
**`ERROR`** The error string

Definition at line 699 of file `generated_mesh_routines.f90`.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `GENERATED_MESH_REGULAR_MESH_TYPE`.

Referenced by `GENERATED_MESH_NUMBER_OF_ELEMENTS_SET_NUMBER()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.24.2.16 `REAL(DP),pointer GENERATED_MESH_ROUTINES::GENERATED_MESH_ORIGIN_GET`** (`TYPE(GENERATED_MESH_TYPE),pointer GENERATED_MESH, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)`)

Get the origin of a generated mesh.

**Parameters:**

**`GENERATED_MESH`** A pointer to the generated mesh to get the type of  
**`ERR`** The error code  
**`ERROR`** The error string

Definition at line 742 of file `generated_mesh_routines.f90`.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `GENERATED_MESH_REGULAR_MESH_TYPE`.

Here is the call graph for this function:

**6.24.2.17 subroutine `GENERATED_MESH_ROUTINES::GENERATED_MESH_ORIGIN_SET_NUMBER`** (`INTEGER(INTG),intent(in) USER_NUMBER, REAL(DP),dimension(:),intent(in) ORIGIN, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, */ [private]`)

Sets/changes the origin of a generated mesh.

Definition at line 779 of file generated\_mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `GENERATED_MESH_ORIGIN_SET_PTR()`,    and    `GENERATED_MESH_USER_NUMBER_FIND()`.

Here is the call graph for this function:

**6.24.2.18 subroutine `GENERATED_MESH_ROUTINES::GENERATED_MESH_ORIGIN_SET_PTR`** (TYPE(GENERATED\_MESH\_TYPE),pointer ***GENERATED\_MESH***,  
***REAL(DP)***,dimension(:),intent(in) ***ORIGIN***, ***INTEGER(INTG)***,intent(out) ***ERR***,  
***TYPE(VARYING\_STRING)***,intent(out) ***ERROR***, \*) [private]

Sets/changes the origin of a generated mesh.

#### Parameters:

***GENERATED\_MESH*** A pointer to the generated mesh to set the type of

***ERR*** The error code

***ERROR*** The error string

Definition at line 810 of file generated\_mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `GENERATED_MESH_REGULAR_MESH_TYPE`.

Referenced by `GENERATED_MESH_ORIGIN_SET_NUMBER()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.24.2.19 subroutine `GENERATED_MESH_ROUTINES::GENERATED_MESH_REGULAR_CREATE_FINISH`** (TYPE(GENERATED\_MESH\_TYPE),pointer  
***GENERATED\_MESH***, ***INTEGER(INTG)***,intent(in) ***MESH\_USER\_NUMBER***,  
***INTEGER(INTG)***,intent(out) ***ERR***, ***TYPE(VARYING\_STRING)***,intent(out) ***ERROR***,  
\*) [private]

Start to create the regular generated mesh type.

#### Parameters:

***GENERATED\_MESH*** A pointer to the generated mesh

***ERR*** The error code

***ERROR*** The error string

Definition at line 856 of file generated\_mesh\_routines.f90.

References    `KINDS::_DP`,    `COORDINATE_ROUTINES::COORDINATE_RECTANGULAR_CARTESIAN_TYPE`,    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `MESH_ROUTINES::MESH_CREATE_FINISH()`, `MESH_ROUTINES::MESH_CREATE_START()`,    `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_FINISH()`,    `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_START()`,    `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_NODES_SET()`,    and    `NODE_ROUTINES::NODES_CREATE_START()`.

Referenced by GENERATED\_MESH\_CREATE\_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.24.2.20 subroutine GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_-  
REGULAR\_FINALISE (TYPE(GENERATED\_MESH\_REGULAR\_TYPE),pointer  
REGULAR\_MESH, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_-  
STRING),intent(out) *ERROR*, \*) [private]**

Finalise the regular mesh type.

#### Parameters:

**REGULAR\_MESH** A pointer to the generated mesh

**ERR** The error code

**ERROR** The error string

Definition at line 1036 of file generated\_mesh\_routines.f90.

References   BASE\_ROUTINES::ENTERS(),   BASE\_ROUTINES::ERRORS(),   and   BASE\_-ROUTINES::EXITS().

Referenced by GENERATED\_MESH\_FINALISE(), and GENERATED\_MESH\_TYPE\_SET\_PTR().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.24.2.21 subroutine GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_REGULAR\_-  
INITIALISE (TYPE(GENERATED\_MESH\_TYPE),pointer *GENERATED\_MESH*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
\*) [private]**

Initialise the regular generated mesh type.

#### Parameters:

**GENERATED\_MESH** A pointer to the generated mesh

**ERR** The error code

**ERROR** The error string

Definition at line 1065 of file generated\_mesh\_routines.f90.

References   BASE\_ROUTINES::ENTERS(),   BASE\_ROUTINES::ERRORS(),   BASE\_-ROUTINES::EXITS(), and GENERATED\_MESH\_REGULAR\_MESH\_TYPE.

Referenced by GENERATED\_MESH\_INITIALISE(), and GENERATED\_MESH\_TYPE\_SET\_PTR().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.24.2.22 INTEGER(INTG) GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_-  
TYPE\_GET (TYPE(GENERATED\_MESH\_TYPE),pointer *GENERATED\_MESH*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*)**

Gets the type of a generated mesh.

**Parameters:**

***GENERATED\_MESH*** A pointer to the generated mesh to set the type of  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 1101 of file generated\_mesh\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    and    **BASE\_-ROUTINES::EXITS()**.

Here is the call graph for this function:

**6.24.2.23 subroutine GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_-  
TYPE\_SET\_NUMBER (INTEGER(INTG),intent(in) *USER\_NUMBER*,  
INTEGER(INTG),intent(in) *GENERATED\_TYPE*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Sets/changes the type of a generated mesh.

Definition at line 1131 of file generated\_mesh\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    **BASE\_-ROUTINES::EXITS()**,    **GENERATED\_MESH\_TYPE\_SET\_PTR()**,    and    **GENERATED\_MESH\_-USER\_NUMBER\_FIND()**.

Here is the call graph for this function:

**6.24.2.24 subroutine GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_TYPE\_-  
SET\_PTR (TYPE(GENERATED\_MESH\_TYPE),pointer *GENERATED\_MESH*,  
INTEGER(INTG),intent(in) *GENERATED\_TYPE*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Sets/changes the type of a generated mesh.

**Parameters:**

***GENERATED\_MESH*** A pointer to the generated mesh to set the type of  
***GENERATED\_TYPE*** The type of mesh to generate

**See also:**

[GENERATED\\_MESH\\_ROUTINES::GeneratedMeshTypes](#),  
[GENERATED\\_MESH\\_ROUTINES](#)

***ERR*** The error code

***ERROR*** The error string

Definition at line 1162 of file generated\_mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `GENERATED_MESH_REGULAR_FINALISE()`,    `GENERATED_MESH_REGULAR_INITIALISE()`, and `GENERATED_MESH_REGULAR_MESH_TYPE`.

Referenced by `GENERATED_MESH_TYPE_SET_NUMBER()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.24.2.25 subroutine `GENERATED_MESH_ROUTINES::GENERATED_MESH_USER_NUMBER_FIND`** (`INTEGER(INTG),intent(in) USER_NUMBER,`  
`TYPE(GENERATED_MESH_TYPE),pointer GENERATED_MESH,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`  
`*) [private]`

Finds and returns in generated mesh a pointer to that identified by `USER_NUMBER`. If no generated mesh with that number exists left nullified.

#### Parameters:

**`USER_NUMBER`** The user number of the mesh to find

**`GENERATED_MESH`** On return, a pointer to the generated mesh of the specified user number. In no generated mesh with the specified user number exists the pointer is returned NULL.

**`ERR`** The error code

**`ERROR`** The error string

Definition at line 1211 of file `generated_mesh_routines.f90`.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `GENERATED_MESHES`, and `KINDS::PTR`.

Referenced by `GENERATED_MESH_BASIS_SET_NUMBER()`,    `GENERATED_MESH_EXTENT_SET_NUMBER()`,    `GENERATED_MESH_NUMBER_OF_ELEMENTS_SET_NUMBER()`, `GENERATED_MESH_ORIGIN_SET_NUMBER()`,    and    `GENERATED_MESH_TYPE_SET_NUMBER()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.24.2.26 subroutine `GENERATED_MESH_ROUTINES::GENERATED_MESHES_FINALISE`** (`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`  
`*)`

Finalises all generated meshes and deallocates all memory.

#### Parameters:

**`ERR`** The error code

**`ERROR`** The error string

Definition at line 1248 of file `generated_mesh_routines.f90`.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `GENERATED_MESHES`, and `KINDS::PTR`.

Referenced by CMISS::CMISS\_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.24.2.27 subroutine GENERATED\_MESH\_ROUTINES::GENERATED\_MESSES\_INITIALISE (INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Initialises all generated meshes.

**Parameters:**

***ERR*** The error code

***ERROR*** The error string

Definition at line 1275 of file generated\_mesh\_routines.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_ROUTINES::EXITS(), and GENERATED\_MESSES.

Referenced by CMISS::CMISS\_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

### 6.24.3 Variable Documentation

**6.24.3.1 TYPE(GENERATED\_MESSES\_TYPE),target GENERATED\_MESH\_ROUTINES::GENERATED\_MESSES**

Definition at line 76 of file generated\_mesh\_routines.f90.

Referenced by GENERATED\_MESH\_CREATE\_START(), GENERATED\_MESH\_DESTROY\_NUMBER(), GENERATED\_MESH\_DESTROY\_PTR(), GENERATED\_MESH\_USER\_NUMBER\_FIND(), GENERATED\_MESSES\_FINALISE(), and GENERATED\_MESSES\_INITIALISE().

## 6.25 INPUT\_OUTPUT Namespace Reference

This module handles all formating and input and output.

### Classes

- interface [WRITE\\_STRING](#)  
*Write a string to a given output stream.*
- interface [WRITE\\_STRING\\_VALUE](#)  
*Write a string followed by a value to a given output stream.*
- interface [WRITE\\_STRING\\_TWO\\_VALUE](#)  
*Write a string, value, string then a value to a given output stream.*
- interface [WRITE\\_STRING\\_FMT\\_VALUE](#)  
*Write a string followed by a value formatted in a particular way to a specified output stream.*
- interface [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE](#)  
*Write a string, value, string then a value with the values formatted in a particular way to a given output stream.*
- interface [WRITE\\_STRING\\_VECTOR](#)  
*Write a string followed by a vector to a specified output stream.*
- interface [WRITE\\_STRING\\_IDX\\_VECTOR](#)  
*Write a string followed by a indexed vector to a specified output stream.*
- interface [WRITE\\_STRING\\_MATRIX](#)  
*Write a string followed by a matrix to a specified output stream.*

### Functions

- subroutine [WRITE\\_STRING\\_C](#) (ID, STRING, ERR, ERROR,\*)  
*Writes the character STRING to the given output stream specified by ID.*
- subroutine [WRITE\\_STRING\\_VS](#) (ID, STRING, ERR, ERROR,\*)  
*Writes the varying string STRING to the given output stream specified by ID.*
- subroutine [WRITE\\_STRING\\_VALUE\\_C](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)  
*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.*
- subroutine [WRITE\\_STRING\\_VALUE\\_DP](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)  
*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.*
- subroutine [WRITE\\_STRING\\_VALUE\\_INTG](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)

*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.*

- subroutine [WRITE\\_STRING\\_VALUE\\_LINTG](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)  
*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.*
- subroutine [WRITE\\_STRING\\_VALUE\\_L](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)  
*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.*
- subroutine [WRITE\\_STRING\\_VALUE\\_SP](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)  
*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.*
- subroutine [WRITE\\_STRING\\_VALUE\\_VS](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)  
*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.*
- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_C\\_C](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)  
*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_C\\_DP](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)  
*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_C\\_INTG](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)  
*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_C\\_L](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)  
*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_C\\_SP](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)  
*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_C\\_VS](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted varying string SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_DP\\_C](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_DP\\_DP](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_DP\\_INTG](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_DP\\_L](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_DP\\_SP](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_DP\\_VS](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_INTG\\_C](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_INTG\\_DP](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_INTG\\_INTG](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_INTG\\_L](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_INTG\\_SP](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_INTG\\_VS](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_L\\_C](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_L\\_DP](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_L\\_INTG](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_L\\_L](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_L\\_SP](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_L\\_VS](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_SP\\_C](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_SP\\_DP](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_SP\\_INTG](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_SP\\_L](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_SP\\_SP](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_SP\\_VS](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted varying string SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_VS\\_C](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_VS\\_DP](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_VS\\_INTG](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_VS\\_L](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_VS\\_SP](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_TWO\\_VALUE\\_VS\\_VS](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted varying string SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [WRITE\\_STRING\\_FMT\\_VALUE\\_C](#) (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)

*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.*

- subroutine [WRITE\\_STRING\\_FMT\\_VALUE\\_DP](#) (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)

*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.*

- subroutine [WRITE\\_STRING\\_FMT\\_VALUE\\_INTG](#) (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)

*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.*

- subroutine [WRITE\\_STRING\\_FMT\\_VALUE\\_LINTG](#) (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)

*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.*

- subroutine [WRITE\\_STRING\\_FMT\\_VALUE\\_L](#) (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)

*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.*

- subroutine [WRITE\\_STRING\\_FMT\\_VALUE\\_SP](#) (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)

*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.*

- subroutine **WRITE\_STRING\_FMT\_VALUE\_VS** (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)

*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.*

- subroutine **WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted varying string SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WRITE\_STRING\_FMT** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE,&SECOND\_FORMAT, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WRITE\_STRING\_FMT** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE,&SECOND\_FORMAT, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WRITE\_STRING\_FMT** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE,&SECOND\_FORMAT, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [WR](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [WRITE\\_STRING\\_FMT](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE,&SECOND\_FORMAT, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [WRITE\\_STRING\\_FMT](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE,&SECOND\_FORMAT, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [WR](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [WR](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [WR](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [WR](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [WR](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [WR](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WRITE\_STRING\_FMT** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE,&SECOND\_FORMAT, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [WR](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [WRITE\\_STRING\\_FMT](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE,&SECOND\_FORMAT, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [WR](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [WR](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [WR](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted varying string SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [WR](#) (ID, FIRST\_IDX, DELTA, LAST\_IDX, NUMBER\_FIRST, NUMBER\_REPEAT, VECTOR, FIRST\_FORMAT, REPEAT\_FORMAT,&ERR, ERROR,\*)

*Writes the given double precision VECTOR to the given output stream specified by ID. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_IDX and LAST\_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.*

- subroutine [WR](#) (ID, FIRST\_IDX, DELTA, LAST\_IDX, NUMBER\_FIRST, NUMBER\_REPEAT, VECTOR, FIRST\_FORMAT, REPEAT\_FORMAT,&ERR, ERROR,\*)

*Writes the given integer VECTOR to the given output stream specified by ID. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_IDX and LAST\_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.*

- subroutine [WR](#) (ID, FIRST\_IDX, DELTA, LAST\_IDX, NUMBER\_FIRST, NUMBER\_REPEAT, VECTOR, FIRST\_FORMAT, REPEAT\_FORMAT,&ERR, ERROR,\*)

*Writes the given integer VECTOR to the given output stream specified by ID. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_IDX and LAST\_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.*

- subroutine [WR](#) (ID, FIRST\_IDX, DELTA, LAST\_IDX, NUMBER\_FIRST, NUMBER\_REPEAT, VECTOR, FIRST\_FORMAT, REPEAT\_FORMAT,&ERR, ERROR,\*)

*Writes the given logical VECTOR to the given output stream specified by ID. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_IDX and LAST\_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.*

- subroutine [WR](#) (ID, FIRST\_IDX, DELTA, LAST\_IDX, NUMBER\_FIRST, NUMBER\_REPEAT, VECTOR, FIRST\_FORMAT, REPEAT\_FORMAT,&ERR, ERROR,\*)

*Writes the given single precision VECTOR to the given output stream specified by ID. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_IDX and LAST\_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.*

- subroutine [WRITE\\_STRING\\_IDX](#) (ID, NUM\_INDICES, INDICES, DELTA, NUMBER\_FIRST, NUMBER\_REPEAT, VECTOR, FIRST\_FORMAT,&REPEAT\_FORMAT, ERR, ERROR,\*)

*Writes the given indexed double precision VECTOR to the given output stream specified by ID. NUM\_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. DELTA is the number of actual indices to skip for each index.*

- subroutine [WRITE\\_STRING\\_IDX](#) (ID, NUM\_INDICES, INDICES, DELTA, NUMBER\_FIRST, NUMBER\_REPEAT, VECTOR, FIRST\_FORMAT,&REPEAT\_FORMAT, ERR, ERROR,\*)

*Writes the given indexed integer VECTOR to the given output stream specified by ID. NUM\_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. DELTA is the number of actual indices to skip for each index.*

- subroutine [WRITE\\_STRING\\_IDX](#) (ID, NUM\_INDICES, INDICES, DELTA, NUMBER\_FIRST, NUMBER\_REPEAT, VECTOR, FIRST\_FORMAT,&REPEAT\_FORMAT, ERR, ERROR,\*)

*Writes the given indexed integer VECTOR to the given output stream specified by ID. NUM\_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. DELTA is the number of actual indices to skip for each index.*

- subroutine [WRITE\\_STRING\\_IDX](#) (ID, NUM\_INDICES, INDICES, DELTA, NUMBER\_FIRST, NUMBER\_REPEAT, VECTOR, FIRST\_FORMAT,&REPEAT\_FORMAT, ERR, ERROR,\*)

*Writes the given indexed logical VECTOR to the given output stream specified by ID. NUM\_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the*

*number of data items in the REPEAT\_FORMAT. DELTA is the number of actual indices to skip for each index.*

- subroutine [WRITE\\_STRING\\_IDX](#) (ID, NUM\_INDICES, INDICES, DELTA, NUMBER\_FIRST, NUMBER\_REPEAT, VECTOR, FIRST\_FORMAT,&REPEAT\_FORMAT, ERR, ERROR,\*)

*Writes the given indexed single precision VECTOR to the given output stream specified by ID. NUM\_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. DELTA is the number of actual indices to skip for each index.*

- subroutine [WRITE\\_STRING\\_MATRIX\\_DP](#) (ID, FIRST\_ROW, DELTA\_ROW, LAST\_ROW, FIRST\_COLUMN, DELTA\_COLUMN, LAST\_COLUMN, NUMBER\_FIRS,&NUMBER\_REPEAT, MATRIX, INDEX\_FORMAT\_TYPE, MATRIX\_NAME\_FORMAT, ROW\_INDEX\_FORMAT, FIRST\_FORMAT, REPEAT\_FORMAT, ERR, ERROR,\*)

*Writes the given double precision MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX\_FORMAT\_TYPE. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_ONLY then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated named with the FIRST\_FORMAT. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated with ROW\_INDEX\_FORMAT and concatenated with FIRST\_FORMAT. Note that with a WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_ROW/FIRST\_COLUMN and LAST\_ROW/LAST\_COLUMN are the extents of the row/column and DELTA\_ROW/DELTA\_COLUMN is the NUMBER of indices to skip for each row/column index.*

- subroutine [WRITE\\_STRING\\_MATRIX\\_INTG](#) (ID, FIRST\_ROW, DELTA\_ROW, LAST\_ROW, FIRST\_COLUMN, DELTA\_COLUMN, LAST\_COLUMN, NUMBER\_FI ST,&NUMBER\_REPEAT, MATRIX, INDEX\_FORMAT\_TYPE, MATRIX\_NAME\_FORMAT, ROW\_INDEX\_FORMAT, FIRST\_FORMAT, REPEAT\_FORMAT, ERR, ERROR,\*)

*Writes the given integer MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX\_FORMAT\_TYPE. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_ONLY then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated named with the FIRST\_FORMAT. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated with ROW\_INDEX\_FORMAT and concatenated with FIRST\_FORMAT. Note that with a WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_ROW/FIRST\_COLUMN and LAST\_ROW/LAST\_COLUMN are the extents of the row/column and DELTA\_ROW/DELTA\_COLUMN is the NUMBER of indices to skip for each row/column index.*

- subroutine [WRITE\\_STRING\\_MATRIX\\_LINTG](#) (ID, FIRST\_ROW, DELTA\_ROW, LAST\_ROW, FIRST\_COLUMN, DELTA\_COLUMN, LAST\_COLUMN, NUMBER\_F RST,&NUMBER\_REPEAT, MATRIX, INDEX\_FORMAT\_TYPE, MATRIX\_NAME\_FORMAT, ROW\_INDEX\_FORMAT, FIRST\_FORMAT, REPEAT\_FORMAT, ERR, ERROR,\*)
- subroutine [WRITE\\_STRING\\_MATRIX\\_L](#) (ID, FIRST\_ROW, DELTA\_ROW, LAST\_ROW, FIRST\_COLUMN, DELTA\_COLUMN, LAST\_COLUMN, NUMBER\_FIRST &NUMBER\_REPEAT, MATRIX, INDEX\_FORMAT\_TYPE, MATRIX\_NAME\_FORMAT, ROW\_INDEX\_FORMAT, FIRST\_FORMAT, REPEAT\_FORMAT, ERR, ERROR,\*)

*Writes the given logical MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX\_FORMAT\_TYPE. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_ONLY then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated named with the FIRST\_FORMAT. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated with ROW\_INDEX\_FORMAT and concatenated with FIRST\_FORMAT. Note that with a WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_ROW/FIRST\_COLUMN and LAST\_ROW/LAST\_COLUMN are the extents of the row/column and DELTA\_ROW/DELTA\_COLUMN is the NUMBER of indices to skip for each row/column index.*

- subroutine `WRITE_STRING_MATRIX_SP` (ID, FIRST\_ROW, DELTA\_ROW, LAST\_ROW, FIRST\_COLUMN, DELTA\_COLUMN, LAST\_COLUMN, NUMBER\_FIRS,&NUMBER\_REPEAT, MATRIX, INDEX\_FORMAT\_TYPE, MATRIX\_NAME\_FORMAT, ROW\_INDEX\_FORMAT, FIRST\_FORMAT, REPEAT\_FORMAT, ERR, ERROR,\*)

*Writes the given single precision MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX\_FORMAT\_TYPE. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_ONLY then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated named with the FIRST\_FORMAT. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated with ROW\_INDEX\_FORMAT and concatenated with FIRST\_FORMAT. Note that with a WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_ROW/FIRST\_COLUMN and LAST\_ROW/LAST\_COLUMN are the extents of the row/column and DELTA\_ROW/DELTA\_COLUMN is the NUMBER of indices to skip for each row/column index.*

## Variables

- INTEGER(INTG), parameter `WRITE_STRING_MATRIX_NAME_ONLY` = 1

*Write the matrix name with out any indices.*

- INTEGER(INTG), parameter `WRITE_STRING_MATRIX_NAME_AND_INDICES` = 2

*Write the matrix name together with the matrix indices.*

### 6.25.1 Detailed Description

This module handles all formating and input and output.

## 6.25.2 Function Documentation

**6.25.2.1 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
 INTEGER(INTG),intent(in) *FIRST\_IDX*, INTEGER(INTG),intent(in)  
*DELTA*, INTEGER(INTG),intent(in) *LAST\_IDX*, INTEGER(INTG),intent(in)  
*NUMBER\_FIRST*, INTEGER(INTG),intent(in) *NUMBER\_REPEAT*,  
 REAL(SP),dimension(:),intent(in) *VECTOR*, CHARACTER(LEN=\*),intent(in)  
*FIRST\_FORMAT*, CHARACTER(LEN=\*),intent(in) *REPEAT\_FORMAT*, &,intent(out)  
*ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the given single precision VECTOR to the given output stream specified by ID. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_IDX and LAST\_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.

### Parameters:

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_IDX*** The first index of the vector to output

***DELTA*** The delta increment to be used when outputting the first through to the last vector index

***LAST\_IDX*** The last index of the vector to output

***NUMBER\_FIRST*** The number of vector elements to be output on the first line

***NUMBER\_REPEAT*** The number of vector elements to be output on the second and subsequently repeated lines

***VECTOR*** The vector to be output

***FIRST\_FORMAT*** The format string to be used for the first line of output

***REPEAT\_FORMAT*** The format type to be used for the second and subsequently repeated lines of output

Definition at line 3380 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.2 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
 INTEGER(INTG),intent(in) *FIRST\_IDX*, INTEGER(INTG),intent(in)  
*DELTA*, INTEGER(INTG),intent(in) *LAST\_IDX*, INTEGER(INTG),intent(in)  
*NUMBER\_FIRST*, INTEGER(INTG),intent(in) *NUMBER\_REPEAT*,  
 LOGICAL,dimension(:),intent(in) *VECTOR*, CHARACTER(LEN=\*),intent(in)  
*FIRST\_FORMAT*, CHARACTER(LEN=\*),intent(in) *REPEAT\_FORMAT*, &,intent(out)  
*ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the given logical VECTOR to the given output stream specified by ID. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_IDX and LAST\_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_IDX** The first index of the vector to output

**DELTA** The delta increment to be used when outputting the first through to the last vector index

**LAST\_IDX** The last index of the vector to output

**NUMBER\_FIRST** The number of vector elements to be output on the first line

**NUMBER\_REPEAT** The number of vector elements to be output on the second and subsequently repeated lines

**VECTOR** The vector to be output

**FIRST\_FORMAT** The format string to be used for the first line of output

**REPEAT\_FORMAT** The format type to be used for the second and subsequently repeated lines of output

Definition at line 3335 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

```
6.25.2.3 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
INTEGER(INTG),intent(in) FIRST_IDX, INTEGER(INTG),intent(in)
DELTA, INTEGER(INTG),intent(in) LAST_IDX, INTEGER(INTG),intent(in)
NUMBER_FIRST, INTEGER(INTG),intent(in) NUMBER_REPEAT,
INTEGER(LINTG),dimension(:,intent(in) VECTOR, CHARACTER(LEN=*),intent(in)
FIRST_FORMAT, CHARACTER(LEN=*),intent(in) REPEAT_FORMAT, &,intent(out)
ERR, INTEGER(INTG),intent(out) error, *)
```

Writes the given integer VECTOR to the given output stream specified by ID. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_IDX and LAST\_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_IDX** The first index of the vector to output

**DELTA** The delta increment to be used when outputting the first through to the last vector index

**LAST\_IDX** The last index of the vector to output

**NUMBER\_FIRST** The number of vector elements to be output on the first line

**NUMBER\_REPEAT** The number of vector elements to be output on the second and subsequently repeated lines

**VECTOR** The vector to be output

**FIRST\_FORMAT** The format string to be used for the first line of output

**REPEAT\_FORMAT** The format type to be used for the second and subsequently repeated lines of output

Definition at line 3290 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.25.2.4 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
INTEGER(INTG),intent(in) *FIRST\_IDX*, INTEGER(INTG),intent(in)  
*DELTA*, INTEGER(INTG),intent(in) *LAST\_IDX*, INTEGER(INTG),intent(in)  
*NUMBER\_FIRST*, INTEGER(INTG),intent(in) *NUMBER\_REPEAT*,  
INTEGER(INTG),dimension(:,),intent(in) *VECTOR*, CHARACTER(LEN=\*),intent(in)  
*FIRST\_FORMAT*, CHARACTER(LEN=\*),intent(in) *REPEAT\_FORMAT*, &,intent(out)  
*ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the given integer VECTOR to the given output stream specified by ID. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_IDX and LAST\_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.

#### Parameters:

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_IDX*** The first index of the vector to output

***DELTA*** The delta increment to be used when outputing the first through to the last vector index

***LAST\_IDX*** The last index of the vector to output

***NUMBER\_FIRST*** The number of vector elements to be output on the first line

***NUMBER\_REPEAT*** The number of vector elements to be output on the second and subsequently repeated lines

***VECTOR*** The vector to be output

***FIRST\_FORMAT*** The format string to be used for the first line of output

***REPEAT\_FORMAT*** The format type to be used for the second and subsequently repeated lines of output

Definition at line 3245 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

---

**6.25.2.5 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
INTEGER(INTG),intent(in) *FIRST\_IDX*, INTEGER(INTG),intent(in)  
*DELTA*, INTEGER(INTG),intent(in) *LAST\_IDX*, INTEGER(INTG),intent(in)  
*NUMBER\_FIRST*, INTEGER(INTG),intent(in) *NUMBER\_REPEAT*,  
REAL(DP),dimension(:,intent(in) *VECTOR*, CHARACTER(LEN=\*),intent(in)  
*FIRST\_FORMAT*, CHARACTER(LEN=\*),intent(in) *REPEAT\_FORMAT*, &,intent(out)  
*ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the given double precision VECTOR to the given output stream specified by ID. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_IDX and LAST\_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_IDX*** The first index of the vector to output

***DELTA*** The delta increment to be used when outputting the first through to the last vector index

***LAST\_IDX*** The last index of the vector to output

***NUMBER\_FIRST*** The number of vector elements to be output on the first line

***NUMBER\_REPEAT*** The number of vector elements to be output on the second and subsequently repeated lines

***VECTOR*** The vector to be output

***FIRST\_FORMAT*** The format string to be used for the first line of output

***REPEAT\_FORMAT*** The format type to be used for the second and subsequently repeated lines of output

Definition at line 3200 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.6 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, TYPE(VARYING\_-  
STRING),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in)  
*FIRST\_FORMAT*, CHARACTER(LEN=\*),intent(in) *SECOND\_-  
STRING*, TYPE(VARYING\_STRING),intent(in) *SECOND\_VALUE*,  
CHARACTER(LEN=\*),intent(in) *SECOND\_FORMAT*, &,intent(out) *ERR*,  
INTEGER(INTG),intent(out) *error*, \*)**

Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted varying string SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**SECOND\_FORMAT** The format string to be used to format the second value

Definition at line 3166 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.7 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, TYPE(VARYING\_STRING),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *FIRST\_FORMAT*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, REAL(SP),intent(in) *SECOND\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_FORMAT*, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

#### Parameters:

***ID*** The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**SECOND\_FORMAT** The format string to be used to format the second value

Definition at line 3131 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.8 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
 CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, TYPE(VARYING\_-  
 STRING),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in)  
*FIRST\_FORMAT*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*,  
 LOGICAL,intent(in) *SECOND\_VALUE*, CHARACTER(LEN=\*),intent(in)  
*SECOND\_FORMAT*, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the *FIRST\_STRING* followed by a formatted varying string *FIRST\_VALUE* and the *SECOND\_STRING* followed by a formatted logical *SECOND\_VALUE* to the given output stream specified by *ID*. *FIRST\_FORMAT* is used to format the first value and *SECOND\_FORMAT* is used to format the second value.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***FIRST\_FORMAT*** The format string to be used to format the first value

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***SECOND\_FORMAT*** The format string to be used to format the second value

Definition at line 3097 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL\\_TO\\_VSTRING\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.9 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
 CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, TYPE(VARYING\_-  
 STRING),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in)  
*FIRST\_FORMAT*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*,  
 REAL(DP),intent(in) *SECOND\_VALUE*, CHARACTER(LEN=\*),intent(in)  
*SECOND\_FORMAT*, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the *FIRST\_STRING* followed by a formatted varying string *FIRST\_VALUE* and the *SECOND\_STRING* followed by a formatted double precision *SECOND\_VALUE* to the given output stream specified by *ID*. *FIRST\_FORMAT* is used to format the first value and *SECOND\_FORMAT* is used to format the second value.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***FIRST\_FORMAT*** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**SECOND\_FORMAT** The format string to be used to format the second value

Definition at line 3027 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.25.2.10 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
 CHARACTER(LEN=\*)intent(in) *FIRST\_STRING*, TYPE(VARYING\_  
 STRING),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*)intent(in)  
*FIRST\_FORMAT*, CHARACTER(LEN=\*)intent(in) *SECOND\_STRING*,  
 CHARACTER(LEN=\*)intent(in) *SECOND\_VALUE*, CHARACTER(LEN=\*)intent(in)  
*SECOND\_FORMAT*, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the *FIRST\_STRING* followed by a formatted varying string *FIRST\_VALUE* and the *SECOND\_STRING* followed by a formatted character *SECOND\_VALUE* to the given output stream specified by *ID*. *FIRST\_FORMAT* is used to format the first value and *SECOND\_FORMAT* is used to format the second value.

#### Parameters:

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***FIRST\_FORMAT*** The format string to be used to format the first value

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***SECOND\_FORMAT*** The format string to be used to format the second value

Definition at line 2993 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.25.2.11 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
 CHARACTER(LEN=\*)intent(in) *FIRST\_STRING*, REAL(SP),intent(in)  
*FIRST\_VALUE*, CHARACTER(LEN=\*)intent(in) *FIRST\_FORMAT*,  
 CHARACTER(LEN=\*)intent(in) *SECOND\_STRING*, TYPE(VARYING\_  
 STRING),intent(in) *SECOND\_VALUE*, CHARACTER(LEN=\*)intent(in)  
*SECOND\_FORMAT*, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the *FIRST\_STRING* followed by a formatted single precision *FIRST\_VALUE* and the *SECOND\_STRING* followed by a formatted single precision *SECOND\_VALUE* to the given output stream specified by *ID*. *FIRST\_FORMAT* is used to format the first value and *SECOND\_FORMAT* is used to format the second value.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**SECOND\_FORMAT** The format string to be used to format the second value

Definition at line 2958 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

```
6.25.2.12 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
CHARACTER(LEN=*),intent(in) FIRST_STRING, REAL(SP),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
CHARACTER(LEN=*),intent(in) SECOND_STRING, REAL(SP),intent(in)
SECOND_VALUE, CHARACTER(LEN=*),intent(in) SECOND_FORMAT,
&,intent(out) ERR, INTEGER(INTG),intent(out) error, *)
```

Writes the **FIRST\_STRING** followed by a formatted single precision **FIRST\_VALUE** and the **SECOND\_STRING** followed by a formatted single precision **SECOND\_VALUE** to the given output stream specified by **ID**. **FIRST\_FORMAT** is used to format the first value and **SECOND\_FORMAT** is used to format the second value.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**SECOND\_FORMAT** The format string to be used to format the second value

Definition at line 2919 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.13 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
 CHARACTER(LEN=\*)intent(in) *FIRST\_STRING*, REAL(SP),intent(in)  
*FIRST\_VALUE*, CHARACTER(LEN=\*)intent(in) *FIRST\_FORMAT*,  
 CHARACTER(LEN=\*)intent(in) *SECOND\_STRING*, LOGICAL,intent(in)  
*SECOND\_VALUE*, CHARACTER(LEN=\*)intent(in) *SECOND\_FORMAT*,  
 &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***FIRST\_FORMAT*** The format string to be used to format the first value

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***SECOND\_FORMAT*** The format string to be used to format the second value

Definition at line 2883 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL\\_TO\\_VSTRING\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.14 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
 CHARACTER(LEN=\*)intent(in) *FIRST\_STRING*, REAL(SP),intent(in)  
*FIRST\_VALUE*, CHARACTER(LEN=\*)intent(in) *FIRST\_FORMAT*,  
 CHARACTER(LEN=\*)intent(in) *SECOND\_STRING*, REAL(DP),intent(in)  
*SECOND\_VALUE*, CHARACTER(LEN=\*)intent(in) *SECOND\_FORMAT*,  
 &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***FIRST\_FORMAT*** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**SECOND\_FORMAT** The format string to be used to format the second value

Definition at line 2805 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.25.2.15 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) ID,  
 CHARACTER(LEN=\*),intent(in) FIRST\_STRING, REAL(SP),intent(in)  
 FIRST\_VALUE, CHARACTER(LEN=\*),intent(in) FIRST\_  
 FORMAT, CHARACTER(LEN=\*),intent(in) SECOND\_STRING,  
 CHARACTER(LEN=\*),intent(in) SECOND\_VALUE, CHARACTER(LEN=\*),intent(in)  
 SECOND\_FORMAT, &,intent(out) ERR, INTEGER(INTG),intent(out) error, \*)**

Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**SECOND\_FORMAT** The format string to be used to format the second value

Definition at line 2770 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.25.2.16 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) ID,  
 CHARACTER(LEN=\*),intent(in) FIRST\_STRING, LOGICAL,intent(in)  
 FIRST\_VALUE, CHARACTER(LEN=\*),intent(in) FIRST\_FORMAT,  
 CHARACTER(LEN=\*),intent(in) SECOND\_STRING, TYPE(VARYING\_  
 STRING),intent(in) SECOND\_VALUE, CHARACTER(LEN=\*),intent(in)  
 SECOND\_FORMAT, &,intent(out) ERR, INTEGER(INTG),intent(out) error, \*)**

Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**SECOND\_FORMAT** The format string to be used to format the second value

Definition at line 2735 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL\\_TO\\_VSTRING\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.17 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
 CHARACTER(LEN=\*)intent(in) *FIRST\_STRING*, LOGICAL,intent(in)  
*FIRST\_VALUE*, CHARACTER(LEN=\*)intent(in) *FIRST\_FORMAT*,  
 CHARACTER(LEN=\*)intent(in) *SECOND\_STRING*, REAL(SP),intent(in)  
*SECOND\_VALUE*, CHARACTER(LEN=\*)intent(in) *SECOND\_FORMAT*,  
 &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**SECOND\_FORMAT** The format string to be used to format the second value

Definition at line 2698 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL\\_TO\\_VSTRING\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.18 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
 CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, LOGICAL,intent(in)  
*FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *FIRST\_FORMAT*,  
 CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, LOGICAL,intent(in)  
*SECOND\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_FORMAT*,  
 &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***FIRST\_FORMAT*** The format string to be used to format the first value

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***SECOND\_FORMAT*** The format string to be used to format the second value

Definition at line 2660 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL\\_TO\\_VSTRING\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.19 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
 CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, LOGICAL,intent(in)  
*FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *FIRST\_FORMAT*,  
 CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, INTEGER(INTG),intent(in)  
*SECOND\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_FORMAT*,  
 &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***FIRST\_FORMAT*** The format string to be used to format the first value

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***SECOND\_FORMAT*** The format string to be used to format the second value

Definition at line 2623 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `STRINGS::LOGICAL_TO_VSTRING()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.25.2.20 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
 CHARACTER(LEN=\*)intent(in) *FIRST\_STRING*, LOGICAL,intent(in)  
*FIRST\_VALUE*, CHARACTER(LEN=\*)intent(in) *FIRST\_FORMAT*,  
 CHARACTER(LEN=\*)intent(in) *SECOND\_STRING*, REAL(DP),intent(in)  
*SECOND\_VALUE*, CHARACTER(LEN=\*)intent(in) *SECOND\_FORMAT*,  
 &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the *FIRST\_STRING* followed by a formatted logical *FIRST\_VALUE* and the the *SECOND\_STRING* followed by a formatted double precision *SECOND\_VALUE* to the given output stream specified by *ID*. *FIRST\_FORMAT* is used to format the first value and *SECOND\_FORMAT* is used to format the second value.

#### Parameters:

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***FIRST\_FORMAT*** The format string to be used to format the first value

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***SECOND\_FORMAT*** The format string to be used to format the second value

Definition at line 2584 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `STRINGS::LOGICAL_TO_VSTRING()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.25.2.21 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
 CHARACTER(LEN=\*)intent(in) *FIRST\_STRING*, LOGICAL,intent(in)  
*FIRST\_VALUE*, CHARACTER(LEN=\*)intent(in) *FIRST\_FORMAT*,  
 CHARACTER(LEN=\*)intent(in) *SECOND\_STRING*,  
 CHARACTER(LEN=\*)intent(in) *SECOND\_VALUE*, CHARACTER(LEN=\*)intent(in)  
*SECOND\_FORMAT*, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the *FIRST\_STRING* followed by a formatted logical *FIRST\_VALUE* and the the *SECOND\_STRING* followed by a formatted character *SECOND\_VALUE* to the given output stream specified by *ID*. *FIRST\_FORMAT* is used to format the first value and *SECOND\_FORMAT* is used to format the second value.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**SECOND\_FORMAT** The format string to be used to format the second value

Definition at line 2549 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL\\_TO\\_VSTRING\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

```
6.25.2.22 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
CHARACTER(LEN=*)intent(in) FIRST_STRING, INTEGER(INTG),intent(in)
FIRST_VALUE, CHARACTER(LEN=*)intent(in) FIRST_FORMAT,
CHARACTER(LEN=*)intent(in) SECOND_STRING, LOGICAL,intent(in)
SECOND_VALUE, CHARACTER(LEN=*)intent(in) SECOND_FORMAT,
&,intent(out) ERR, INTEGER(INTG),intent(out) error, *)
```

Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**SECOND\_FORMAT** The format string to be used to format the second value

Definition at line 2437 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL\\_TO\\_VSTRING\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.23 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, INTEGER(INTG),intent(in)  
*FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *FIRST\_FORMAT*,  
CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*,  
CHARACTER(LEN=\*),intent(in) *SECOND\_VALUE*, CHARACTER(LEN=\*),intent(in)  
*SECOND\_FORMAT*, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***FIRST\_FORMAT*** The format string to be used to format the first value

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***SECOND\_FORMAT*** The format string to be used to format the second value

Definition at line 2324 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.24 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, REAL(DP),intent(in)  
*FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *FIRST\_FORMAT*,  
CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, TYPE(VARYING\_-  
*STRING*),intent(in) *SECOND\_VALUE*, CHARACTER(LEN=\*),intent(in)  
*SECOND\_FORMAT*, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***FIRST\_FORMAT*** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**SECOND\_FORMAT** The format string to be used to format the second value

Definition at line 2289 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.25 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
CHARACTER(LEN=*),intent(in) FIRST_STRING, REAL(DP),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
CHARACTER(LEN=*),intent(in) SECOND_STRING, REAL(SP),intent(in)
SECOND_VALUE, CHARACTER(LEN=*),intent(in) SECOND_FORMAT,
&,intent(out) ERR, INTEGER(INTG),intent(out) error, *)
```

Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**SECOND\_FORMAT** The format string to be used to format the second value

Definition at line 2250 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.26 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
CHARACTER(LEN=*),intent(in) FIRST_STRING, REAL(DP),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
CHARACTER(LEN=*),intent(in) SECOND_STRING, LOGICAL,intent(in)
SECOND_VALUE, CHARACTER(LEN=*),intent(in) SECOND_FORMAT,
&,intent(out) ERR, INTEGER(INTG),intent(out) error, *)
```

Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**SECOND\_FORMAT** The format string to be used to format the second value

Definition at line 2212 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL\\_TO\\_VSTRING\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

```
6.25.2.27 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
CHARACTER(LEN=*),intent(in) FIRST_STRING, REAL(DP),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
CHARACTER(LEN=*),intent(in) SECOND_STRING, REAL(DP),intent(in)
SECOND_VALUE, CHARACTER(LEN=*),intent(in) SECOND_FORMAT,
&,intent(out) ERR, INTEGER(INTG),intent(out) error, *)
```

Writes the **FIRST\_STRING** followed by a formatted double precision **FIRST\_VALUE** and the **SECOND\_STRING** followed by a formatted double precision **SECOND\_VALUE** to the given output stream specified by **ID**. **FIRST\_FORMAT** is used to format the first value and **SECOND\_FORMAT** is used to format the second value.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**SECOND\_FORMAT** The format string to be used to format the second value

Definition at line 2134 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.28 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
 CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, REAL(DP),intent(in)  
*FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *FIRST\_FORMAT*,  
 CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*,  
 CHARACTER(LEN=\*),intent(in) *SECOND\_VALUE*, CHARACTER(LEN=\*),intent(in)  
*SECOND\_FORMAT*, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***FIRST\_FORMAT*** The format string to be used to format the first value

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***SECOND\_FORMAT*** The format string to be used to format the second value

Definition at line 2099 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.29 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
 CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, CHARACTER(LEN=\*),intent(in)  
*FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *FIRST\_FORMAT*,  
 CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, TYPE(VARYING\_-  
*STRING*),intent(in) *SECOND\_VALUE*, CHARACTER(LEN=\*),intent(in)  
*SECOND\_FORMAT*, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted varying string SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***FIRST\_FORMAT*** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**SECOND\_FORMAT** The format string to be used to format the second value

Definition at line 2065 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.30 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
CHARACTER(LEN=*),intent(in) FIRST_STRING, CHARACTER(LEN=*),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
CHARACTER(LEN=*),intent(in) SECOND_STRING, REAL(SP),intent(in)
SECOND_VALUE, CHARACTER(LEN=*),intent(in) SECOND_FORMAT,
&,intent(out) ERR, INTEGER(INTG),intent(out) error, *)
```

Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**SECOND\_FORMAT** The format string to be used to format the second value

Definition at line 2030 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.31 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
CHARACTER(LEN=*),intent(in) FIRST_STRING, CHARACTER(LEN=*),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
CHARACTER(LEN=*),intent(in) SECOND_STRING, LOGICAL,intent(in)
SECOND_VALUE, CHARACTER(LEN=*),intent(in) SECOND_FORMAT,
&,intent(out) ERR, INTEGER(INTG),intent(out) error, *)
```

Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**SECOND\_FORMAT** The format string to be used to format the second value

Definition at line 1996 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL\\_TO\\_VSTRING\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

```
6.25.2.32 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
CHARACTER(LEN=*),intent(in) FIRST_STRING, CHARACTER(LEN=*),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
CHARACTER(LEN=*),intent(in) SECOND_STRING, INTEGER(INTG),intent(in)
SECOND_VALUE, CHARACTER(LEN=*),intent(in) SECOND_FORMAT,
&,intent(out) ERR, INTEGER(INTG),intent(out) error, *)
```

Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**SECOND\_FORMAT** The format string to be used to format the second value

Definition at line 1961 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.33 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
 CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, CHARACTER(LEN=\*),intent(in)  
*FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *FIRST\_FORMAT*,  
 CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, REAL(DP),intent(in)  
*SECOND\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_FORMAT*,  
 &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***FIRST\_FORMAT*** The format string to be used to format the first value

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***SECOND\_FORMAT*** The format string to be used to format the second value

Definition at line 1926 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.34 subroutine INPUT\_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,  
 CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, CHARACTER(LEN=\*),intent(in)  
*FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *FIRST\_FORMAT*,  
 CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*,  
 CHARACTER(LEN=\*),intent(in) *SECOND\_VALUE*, CHARACTER(LEN=\*),intent(in)  
*SECOND\_FORMAT*, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, \*)**

Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***FIRST\_FORMAT*** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**SECOND\_FORMAT** The format string to be used to format the second value

Definition at line 1892 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

#### 6.25.2.35 subroutine INPUT\_OUTPUT::WRITE\_STRING\_C (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *STRING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Writes the character *STRING* to the given output stream specified by *ID*.

##### Parameters:

***ID*** The ID of the output stream to write to

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***STRING*** The string to write

***ERR*** The error code

***ERROR*** The error string

Definition at line 221 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

#### 6.25.2.36 subroutine INPUT\_OUTPUT::WRITE\_STRING\_FMT (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, TYPE(VARYING\_STRING),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *FIRST\_FORMAT*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, INTEGER(INTG),intent(in) *SECOND\_VALUE*, &,intent(in) *SECOND\_FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Writes the *FIRST\_STRING* followed by a formatted varying string *FIRST\_VALUE* and the *SECOND\_STRING* followed by a formatted integer *SECOND\_VALUE* to the given output stream specified by *ID*. *FIRST\_FORMAT* is used to format the first value and *SECOND\_FORMAT* is used to format the second value.

##### Parameters:

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

**FIRST\_VALUE** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 3062 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.37 subroutine INPUT_OUTPUT::WRITE_STRING_FMT (INTEGER(INTG),intent(in)
ID, CHARACTER(LEN=*),intent(in) FIRST_STRING, REAL(SP),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
CHARACTER(LEN=*),intent(in) SECOND_STRING, INTEGER(INTG),intent(in)
SECOND_VALUE, &,intent(in) SECOND_FORMAT, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

#### See also:

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 2844 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

---

**6.25.2.38 subroutine INPUT\_OUTPUT::WRITE\_STRING\_FMT (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, INTEGER(INTG),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *FIRST\_FORMAT*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, TYPE(VARYING\_-  
STRING),intent(in) *SECOND\_VALUE*, &,intent(in) *SECOND\_FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 2514 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.39 subroutine INPUT\_OUTPUT::WRITE\_STRING\_FMT (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, INTEGER(INTG),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *FIRST\_FORMAT*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, REAL(SP),intent(in) *SECOND\_VALUE*, &,intent(in) *SECOND\_FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 2475 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.40 subroutine INPUT_OUTPUT::WRITE_STRING_FMT (INTEGER(INTG),intent(in)
 ID, CHARACTER(LEN=*),intent(in) FIRST_STRING, INTEGER(INTG),intent(in)
 FIRST_VALUE, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
 CHARACTER(LEN=*),intent(in) SECOND_STRING, INTEGER(INTG),intent(in)
 SECOND_VALUE, &,intent(in) SECOND_FORMAT, INTEGER(INTG),intent(out)
 ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

#### See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 2398 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

---

**6.25.2.41 subroutine INPUT\_OUTPUT::WRITE\_STRING\_FMT (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, INTEGER(INTG),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *FIRST\_FORMAT*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, REAL(DP),intent(in) *SECOND\_VALUE*, &,intent(in) *SECOND\_FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***FIRST\_FORMAT*** The format string to be used to format the first value

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 2359 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.42 subroutine INPUT\_OUTPUT::WRITE\_STRING\_FMT (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, REAL(DP),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *FIRST\_FORMAT*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, INTEGER(INTG),intent(in) *SECOND\_VALUE*, &,intent(in) *SECOND\_FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

**FIRST\_FORMAT** The format string to be used to format the first value

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 2173 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.25.2.43 subroutine INPUT\_OUTPUT::WRITE\_STRING\_FMT\_VALUE\_C**  
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING, CHARACTER(LEN=*),intent(in) VALUE, CHARACTER(LEN=*),intent(in) FORMAT_STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**VALUE** The value to be output

**FORMAT\_STRING** The format string to be used to format the value

**ERR** The error code

**ERROR** The error string

Definition at line 1680 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.25.2.44 subroutine INPUT\_OUTPUT::WRITE\_STRING\_FMT\_VALUE\_DP**  
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING, REAL(DP),intent(in) VALUE, CHARACTER(LEN=*),intent(in) FORMAT_STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**VALUE** The value to be output

**FORMAT\_STRING** The format string to be used to format the value

**ERR** The error code

**ERROR** The error string

Definition at line 1710 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.45 subroutine INPUT\_OUTPUT::WRITE\_STRING\_FMT\_VALUE\_INTG**  
**(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=\*),intent(in) FIRST\_STRING,**  
**INTEGER(INTG),intent(in) VALUE, CHARACTER(LEN=\*),intent(in) FORMAT\_**  
**STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out)**  
**ERROR, \*) [private]**

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**VALUE** The value to be output

**FORMAT\_STRING** The format string to be used to format the value

**ERR** The error code

**ERROR** The error string

Definition at line 1741 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.46 subroutine INPUT\_OUTPUT::WRITE\_STRING\_FMT\_VALUE\_L**  
**(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=\*),intent(in) FIRST\_STRING,**  
**LOGICAL,intent(in) VALUE, CHARACTER(LEN=\*),intent(in) FORMAT\_STRING,**  
**INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,**  
**\*) [private]**

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**VALUE** The value to be output

**FORMAT\_STRING** The format string to be used to format the value

**ERR** The error code

**ERROR** The error string

Definition at line 1801 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL\\_TO\\_VSTRING\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.47 subroutine INPUT\_OUTPUT::WRITE\_STRING\_FMT\_VALUE\_LINTG**  
**(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=\*),intent(in) FIRST\_STRING,**  
**INTEGER(LINTG),intent(in) VALUE, CHARACTER(LEN=\*),intent(in) FORMAT\_**  
**STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out)**  
**ERROR, \*) [private]**

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**VALUE** The value to be output

**FORMAT\_STRING** The format string to be used to format the value

**ERR** The error code

**ERROR** The error string

Definition at line 1771 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.48 subroutine INPUT\_OUTPUT::WRITE\_STRING\_FMT\_VALUE\_SP**  
**(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=\*),intent(in) FIRST\_STRING,**  
**REAL(SP),intent(in) VALUE, CHARACTER(LEN=\*),intent(in) FORMAT\_STRING,**  
**INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,**  
**\*) [private]**

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**VALUE** The value to be output

**FORMAT\_STRING** The format string to be used to format the value

**ERR** The error code

**ERROR** The error string

Definition at line 1831 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

```
6.25.2.49 subroutine INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_VS
 (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in)
 FIRST_STRING, TYPE(VARYING_STRING),intent(in) VALUE,
 CHARACTER(LEN=*),intent(in) FORMAT_STRING, INTEGER(INTG),intent(out)
 ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**VALUE** The value to be output

**FORMAT\_STRING** The format string to be used to format the value

**ERR** The error code

**ERROR** The error string

Definition at line 1862 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.50 subroutine INPUT\_OUTPUT::WRITE\_STRING\_IDX** (INTEGER(INTG),intent(in) *ID*,  
 INTEGER(INTG),intent(in) *NUM\_INDICES*, INTEGER(INTG),dimension(*num\_-  
 indices*),intent(in) *INDICES*, INTEGER(INTG),intent(in) *DELTA*,  
 INTEGER(INTG),intent(in) *NUMBER\_FIRST*, INTEGER(INTG),intent(in)  
*NUMBER\_REPEAT*, REAL(SP),dimension(:),intent(in) *VECTOR*,  
 CHARACTER(LEN=\*),intent(in) *FIRST\_FORMAT*, &,intent(in) *REPEAT\_FORMAT*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Writes the given indexed single precision VECTOR to the given output stream specified by ID. NUM\_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. DELTA is the number of actual indices to skip for each index.

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***NUM\_INDICES*** The number of indices of the vector to output

***INDICES*** INDICES(i). The i'th index of the vector to output

***DELTA*** The delta increment to be used when outputting the first through to the last vector index

***NUMBER\_FIRST*** The number of vector elements to be output on the first line

***NUMBER\_REPEAT*** The number of vector elements to be output on the second and subsequently repeated lines

***VECTOR*** The vector to be output

***FIRST\_FORMAT*** The format string to be used for the first line of output

***ERR*** The error code

***ERROR*** The error string

Definition at line 3610 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.51 subroutine INPUT\_OUTPUT::WRITE\_STRING\_IDX** (INTEGER(INTG),intent(in) *ID*,  
 INTEGER(INTG),intent(in) *NUM\_INDICES*, INTEGER(INTG),dimension(*num\_-  
 indices*),intent(in) *INDICES*, INTEGER(INTG),intent(in) *DELTA*,  
 INTEGER(INTG),intent(in) *NUMBER\_FIRST*, INTEGER(INTG),intent(in)  
*NUMBER\_REPEAT*, LOGICAL,dimension(:),intent(in) *VECTOR*,  
 CHARACTER(LEN=\*),intent(in) *FIRST\_FORMAT*, &,intent(in) *REPEAT\_FORMAT*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Writes the given indexed logical VECTOR to the given output stream specified by ID. NUM\_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST\_FORMAT

is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. DELTA is the number of actual indices to skip for each index.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**NUM\_INDICES** The number of indices of the vector to output

**INDICES** INDICES(i). The i'th index of the vector to output

**DELTA** The delta increment to be used when outputting the first through to the last vector index

**NUMBER\_FIRST** The number of vector elements to be output on the first line

**NUMBER\_REPEAT** The number of vector elements to be output on the second and subsequently repeated lines

**VECTOR** The vector to be output

**FIRST\_FORMAT** The format string to be used for the first line of output

**ERR** The error code

**ERROR** The error string

Definition at line 3565 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

```
6.25.2.52 subroutine INPUT_OUTPUT::WRITE_STRING_IDX (INTEGER(INTG),intent(in) ID,
 INTEGER(INTG),intent(in) NUM_INDICES, INTEGER(INTG),dimension(num_indices),intent(in)
 INDICES, INTEGER(INTG),intent(in) DELTA,
 INTEGER(INTG),intent(in) NUMBER_FIRST, INTEGER(INTG),intent(in)
 NUMBER_REPEAT, INTEGER(LINTG),dimension(:),intent(in) VECTOR,
 CHARACTER(LEN=*),intent(in) FIRST_FORMAT, &,intent(in) REPEAT_FORMAT,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *) [private]
```

Writes the given indexed integer VECTOR to the given output stream specified by ID. NUM\_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. DELTA is the number of actual indices to skip for each index.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**NUM\_INDICES** The number of indices of the vector to output

**INDICES** INDICES(i). The i'th index of the vector to output  
**DELTA** The delta increment to be used when outputting the first through to the last vector index  
**NUMBER\_FIRST** The number of vector elements to be output on the first line  
**NUMBER\_REPEAT** The number of vector elements to be output on the second and subsequently repeated lines  
**VECTOR** The vector to be output  
**FIRST\_FORMAT** The format string to be used for the first line of output  
**ERR** The error code  
**ERROR** The error string

Definition at line 3515 of file input\_output.f90.

References BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::OP\_STRING, and BASE\_ROUTINES::WRITE\_STR().

Here is the call graph for this function:

```
6.25.2.53 subroutine INPUT_OUTPUT::WRITE_STRING_IDX (INTEGER(INTG),intent(in) ID,
 INTEGER(INTG),intent(in) NUM_INDICES, INTEGER(INTG),dimension(num_-
 indices),intent(in) INDICES, INTEGER(INTG),intent(in) DELTA,
 INTEGER(INTG),intent(in) NUMBER_FIRST, INTEGER(INTG),intent(in)
 NUMBER_REPEAT, INTEGER(INTG),dimension(:,intent(in) VECTOR,
 CHARACTER(LEN=*),intent(in) FIRST_FORMAT, &,intent(in) REPEAT_FORMAT,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *) [private]
```

Writes the given indexed integer VECTOR to the given output stream specified by ID. NUM\_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. DELTA is the number of actual indices to skip for each index.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**NUM\_INDICES** The number of indices of the vector to output

**INDICES** INDICES(i). The i'th index of the vector to output

**DELTA** The delta increment to be used when outputting the first through to the last vector index

**NUMBER\_FIRST** The number of vector elements to be output on the first line

**NUMBER\_REPEAT** The number of vector elements to be output on the second and subsequently repeated lines

**VECTOR** The vector to be output

**FIRST\_FORMAT** The format string to be used for the first line of output

**ERR** The error code

**ERROR** The error string

Definition at line 3470 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.54 subroutine INPUT_OUTPUT::WRITE_STRING_IDX (INTEGER(INTG),intent(in) ID,
 INTEGER(INTG),intent(in) NUM_INDICES, INTEGER(INTG),dimension(num_-
 indices),intent(in) INDICES, INTEGER(INTG),intent(in) DELTA,
 INTEGER(INTG),intent(in) NUMBER_FIRST, INTEGER(INTG),intent(in)
 NUMBER_REPEAT, REAL(DP),dimension(:),intent(in) VECTOR,
 CHARACTER(LEN=*),intent(in) FIRST_FORMAT, &,intent(in) REPEAT_FORMAT,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *) [private]
```

Writes the given indexed double precision VECTOR to the given output stream specified by ID. *NUM\_INDICES* is the number of indices and *INDICES*(i) contain the indices of the vector to write. The *FIRST\_FORMAT* is the format initially used, followed by the *REPEAT\_FORMAT* which is repeated as many times as necessary. *NUMBER\_FIRST* is the number of data items in the *FIRST\_FORMAT* and *NUMBER\_REPEAT* is the number of data items in the *REPEAT\_FORMAT*. *DELTA* is the number of actual indices to skip for each index.

#### Parameters:

***ID*** The ID of the output stream. An ID of > 9 specifies file output

#### See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***NUM\_INDICES*** The number of indices of the vector to output

***INDICES*** *INDICES*(i). The i'th index of the vector to output

***DELTA*** The delta increment to be used when outputting the first through to the last vector index

***NUMBER\_FIRST*** The number of vector elements to be output on the first line

***NUMBER\_REPEAT*** The number of vector elements to be output on the second and subsequently repeated lines

***VECTOR*** The vector to be output

***FIRST\_FORMAT*** The format string to be used for the first line of output

***ERR*** The error code

***ERROR*** The error string

Definition at line 3425 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

---

**6.25.2.55 subroutine INPUT\_OUTPUT::WRITE\_STRING\_MATRIX\_DP**

```
(INTEGER(INTG),intent(in) ID, INTEGER(INTG),intent(in) FIRST_ROW,
INTEGER(INTG),intent(in) DELTA_ROW, INTEGER(INTG),intent(in)
LAST_ROW, INTEGER(INTG),intent(in) FIRST_COLUMN,
INTEGER(INTG),intent(in) DELTA_COLUMN, INTEGER(INTG),intent(in)
LAST_COLUMN, NUMBER_FIRS, &,intent(in) NUMBER_REPEAT,
INTEGER(INTG),dimension(:, :, intent(in) matrix, INTEGER(INTG),intent(in)
INDEX_FORMAT_TYPE, CHARACTER(LEN=*=*),intent(in)
MATRIX_NAME_FORMAT, CHARACTER(LEN=*=*),intent(in)
ROW_INDEX_FORMAT, CHARACTER(LEN=*=*),intent(in) FIRST_FORMAT,
CHARACTER(LEN=*=*),intent(in) REPEAT_FORMAT, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the given double precision MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX\_FORMAT\_TYPE. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_ONLY then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated named with the FIRST\_FORMAT. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated with ROW\_INDEX\_FORMAT and concatenated with FIRST\_FORMAT. Note that with a WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_ROW/FIRST\_COLUMN and LAST\_ROW/LAST\_COLUMN are the extents of the row/column and DELTA\_ROW/DELTA\_COLUMN is the NUMBER of indices to skip for each row/column index.

#### Parameters:

***ID*** The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_ROW*** The first row of the matrix to be output

***DELTA\_ROW*** The delta row increment to be used when outputting the first through to the last matrix row

***LAST\_ROW*** The last row of the matrix to be output

***FIRST\_COLUMN*** The first column of the matrix to be output

***DELTA\_COLUMN*** The delta column increase to be used when outputting the first through to the last matrix column

***LAST\_COLUMN*** The last column of the matrix to be output

***INDEX\_FORMAT\_TYPE*** The format type to be used for the matrix name and indices

See also:

[INPUT\\_OUTPUT::MatrixNameIndexFormat](#), [INPUT\\_OUTPUT::MatrixNameIndexFormat](#)

***MATRIX\_NAME\_FORMAT*** The format string to be used to format the matrix name

***ROW\_INDEX\_FORMAT*** The format string to be used to format the row indices

***FIRST\_FORMAT*** The format string to be used for the first line of output

***REPEAT\_FORMAT*** The format type to be used for the second and subsequently repeated lines of output

***ERR*** The error code

**ERROR** The error string

Definition at line 3655 of file input\_output.f90.

References    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::OP_STRING`,    `BASE_ROUTINES::WRITE_STR()`,    `WRITE_STRING_MATRIX_NAME_AND_INDICES`, and `WRITE_STRING_MATRIX_NAME_ONLY`.

Here is the call graph for this function:

```
6.25.2.56 subroutine INPUT_OUTPUT::WRITE_STRING_MATRIX_INTG
 (INTEGER(INTG),intent(in) ID, INTEGER(INTG),intent(in) FIRST_ROW,
 INTEGER(INTG),intent(in) DELTA_ROW, INTEGER(INTG),intent(in)
 LAST_ROW, INTEGER(INTG),intent(in) FIRST_COLUMN,
 INTEGER(INTG),intent(in) DELTA_COLUMN, INTEGER(INTG),intent(in)
 LAST_COLUMN, NUMBER_FIRST, &,intent(in) NUMBER_REPEAT,
 INTEGER(INTG),dimension(:, :, :),intent(in) matrix, INTEGER(INTG),intent(in)
 INDEX_FORMAT_TYPE, CHARACTER(LEN=*=*),intent(in)
 MATRIX_NAME_FORMAT, CHARACTER(LEN=*=*),intent(in)
 ROW_INDEX_FORMAT, CHARACTER(LEN=*=*),intent(in) FIRST_FORMAT,
 CHARACTER(LEN=*=*),intent(in) REPEAT_FORMAT, INTEGER(INTG),intent(out)
 ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the given integer MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX\_FORMAT\_TYPE. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_ONLY then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated named with the FIRST\_FORMAT. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated with ROW\_INDEX\_FORMAT and concatenated with FIRST\_FORMAT. Note that with a WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_ROW/FIRST\_COLUMN and LAST\_ROW/LAST\_COLUMN are the extents of the row/column and DELTA\_ROW/DELTA\_COLUMN is the NUMBER of indices to skip for each row/column index.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

**FIRST\_ROW** The first row of the matrix to be output

**DELTA\_ROW** The delta row increment to be used when outputting the first through to the last matrix row

**LAST\_ROW** The last row of the matrix to be output

**FIRST\_COLUMN** The first column of the matrix to be output

**DELTA\_COLUMN** The delta column increase to be used when outputting the first through to the last matrix column

**LAST\_COLUMN** The last column of the matrix to be output

**INDEX\_FORMAT\_TYPE** The format type to be used for the matrix name and indices

**See also:**

[INPUT\\_OUTPUT::MatrixNameIndexFormat](#), [INPUT\\_OUTPUT::MatrixNameIndexFormat MATRIX\\_NAME\\_FORMAT](#) The format string to be used to format the matrix name  
[INPUT\\_OUTPUT::MatrixNameIndexFormat ROW\\_INDEX\\_FORMAT](#) The format string to be used to format the row indices  
[INPUT\\_OUTPUT::MatrixNameIndexFormat FIRST\\_FORMAT](#) The format string to be used for the first line of output  
[INPUT\\_OUTPUT::MatrixNameIndexFormat REPEAT\\_FORMAT](#) The format type to be used for the second and subsequently repeated lines of output  
[ERR](#) The error code  
[ERROR](#) The error string

Definition at line 3720 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), [BASE\\_ROUTINES::WRITE\\_STR\(\)](#), [WRITE\\_STRING\\_MATRIX\\_NAME\\_AND\\_INDICES](#), and [WRITE\\_STRING\\_MATRIX\\_NAME\\_ONLY](#).

Here is the call graph for this function:

```
6.25.2.57 subroutine INPUT_OUTPUT::WRITE_STRING_MATRIX_L
 (INTEGER(INTG),intent(in) ID, INTEGER(INTG),intent(in) FIRST_ROW,
 INTEGER(INTG),intent(in) DELTA_ROW, INTEGER(INTG),intent(in)
 LAST_ROW, INTEGER(INTG),intent(in) FIRST_COLUMN,
 INTEGER(INTG),intent(in) DELTA_COLUMN, INTEGER(INTG),intent(in)
 LAST_COLUMN, NUMBER_FIRST &,intent(in) NUMBER_REPEAT,
 INTEGER(INTG),dimension(:, :, intent(in) matrix, INTEGER(INTG),intent(in)
 INDEX_FORMAT_TYPE, CHARACTER(LEN=*=*),intent(in)
 MATRIX_NAME_FORMAT, CHARACTER(LEN=*=*),intent(in)
 ROW_INDEX_FORMAT, CHARACTER(LEN=*=*),intent(in) FIRST_FORMAT,
 CHARACTER(LEN=*=*),intent(in) REPEAT_FORMAT, INTEGER(INTG),intent(out)
 ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the given logical MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX\_FORMAT\_TYPE. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_ONLY then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated named with the FIRST\_FORMAT. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated with ROW\_INDEX\_FORMAT and concatenated with FIRST\_FORMAT. Note that with a WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_ROW/FIRST\_COLUMN and LAST\_ROW/LAST\_COLUMN are the extents of the row/column and DELTA\_ROW/DELTA\_COLUMN is the NUMBER of indices to skip for each row/column index.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_ROW** The first row of the matrix to be output

**DELTA\_ROW** The delta row increment to be used when outputting the first through to the last matrix row

**LAST\_ROW** The last row of the matrix to be output

**FIRST\_COLUMN** The first column of the matrix to be output

**DELTA\_COLUMN** The delta column increase to be used when outputting the first through to the last matrix column

**LAST\_COLUMN** The last column of the matrix to be output

**INDEX\_FORMAT\_TYPE** The format type to be used for the matrix name and indices

See also:

[INPUT\\_OUTPUT::MatrixNameIndexFormat](#), [INPUT\\_OUTPUT::MatrixNameIndexFormat](#)

**MATRIX\_NAME\_FORMAT** The format string to be used to format the matrix name

**ROW\_INDEX\_FORMAT** The format string to be used to format the row indices

**FIRST\_FORMAT** The format string to be used for the first line of output

**REPEAT\_FORMAT** The format type to be used for the second and subsequently repeated lines of output

**ERR** The error code

**ERROR** The error string

Definition at line 3857 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, `BASE_ROUTINES::WRITE_STR()`, `WRITE_STRING_MATRIX_NAME_AND_INDICES`, and `WRITE_STRING_MATRIX_NAME_ONLY`.

Here is the call graph for this function:

```
6.25.2.58 subroutine INPUT_OUTPUT::WRITE_STRING_MATRIX_LINTG
 (INTEGER(INTG),intent(in) ID, INTEGER(INTG),intent(in) FIRST_ROW,
 INTEGER(INTG),intent(in) DELTA_ROW, INTEGER(INTG),intent(in)
 LAST_ROW, INTEGER(INTG),intent(in) FIRST_COLUMN,
 INTEGER(INTG),intent(in) DELTA_COLUMN, INTEGER(INTG),intent(in)
 LAST_COLUMN, NUMBER_F_RST, &,intent(in) NUMBER_REPEAT,
 INTEGER(INTG),dimension(:, :, intent(in) matrix, INTEGER(INTG),intent(in)
 INDEX_FORMAT_TYPE, CHARACTER(LEN=*),intent(in)
 MATRIX_NAME_FORMAT, CHARACTER(LEN=*),intent(in)
 ROW_INDEX_FORMAT, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
 CHARACTER(LEN=*),intent(in) REPEAT_FORMAT, INTEGER(INTG),intent(out)
 ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the given long integer MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX\_FORMAT\_TYPE. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_ONLY then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated named with the FIRST\_FORMAT. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated with ROW\_INDEX\_FORMAT and concatenated with FIRST\_FORMAT. Note that with a WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_ROW/FIRST\_COLUMN and LAST\_ROW/LAST\_COLUMN are the

extents of the row/column and DELTA\_ROW/DELTA\_COLUMN is the NUMBER of indices to skip for each row/column index.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_ROW** The first row of the matrix to be output

**DELTA\_ROW** The delta row increment to be used when outputting the first through to the last matrix row

**LAST\_ROW** The last row of the matrix to be output

**FIRST\_COLUMN** The first column of the matrix to be output

**DELTA\_COLUMN** The delta column increase to be used when outputting the first through to the last matrix column

**LAST\_COLUMN** The last column of the matrix to be output

**INDEX\_FORMAT\_TYPE** The format type to be used for the matrix name and indices

See also:

[INPUT\\_OUTPUT::MatrixNameIndexFormat](#), [INPUT\\_OUTPUT::MatrixNameIndexFormat](#)

**MATRIX\_NAME\_FORMAT** The format string to be used to format the matrix name

**ROW\_INDEX\_FORMAT** The format string to be used to format the row indices

**FIRST\_FORMAT** The format string to be used for the first line of output

**REPEAT\_FORMAT** The format type to be used for the second and subsequently repeated lines of output

**ERR** The error code

**ERROR** The error string

Definition at line 3792 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), [BASE\\_ROUTINES::WRITE\\_STR\(\)](#), [WRITE\\_STRING\\_MATRIX\\_NAME\\_AND\\_INDICES](#), and [WRITE\\_STRING\\_MATRIX\\_NAME\\_ONLY](#).

Here is the call graph for this function:

```
6.25.2.59 subroutine INPUT_OUTPUT::WRITE_STRING_MATRIX_SP
 (INTEGER(INTG),intent(in) ID, INTEGER(INTG),intent(in) FIRST_ROW,
 INTEGER(INTG),intent(in) DELTA_ROW, INTEGER(INTG),intent(in)
 LAST_ROW, INTEGER(INTG),intent(in) FIRST_COLUMN,
 INTEGER(INTG),intent(in) DELTA_COLUMN, INTEGER(INTG),intent(in)
 LAST_COLUMN, NUMBER_FIRS, &,intent(in) NUMBER_REPEAT,
 INTEGER(INTG),dimension(:, :, :),intent(in) matrix, INTEGER(INTG),intent(in)
 INDEX_FORMAT_TYPE, CHARACTER(LEN=*=*),intent(in)
 MATRIX_NAME_FORMAT, CHARACTER(LEN=*=*),intent(in)
 ROW_INDEX_FORMAT, CHARACTER(LEN=*=*),intent(in) FIRST_FORMAT,
 CHARACTER(LEN=*=*),intent(in) REPEAT_FORMAT, INTEGER(INTG),intent(out)
 ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the given single precision MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX\_FORMAT\_TYPE. If INDEX\_FORMAT\_TYPE is WRITE\_-

STRING\_MATRIX\_NAME\_ONLY then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated named with the FIRST\_FORMAT. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated with ROW\_INDEX\_FORMAT and concatenated with FIRST\_FORMAT. Note that with a WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_ROW/FIRST\_COLUMN and LAST\_ROW/LAST\_COLUMN are the extents of the row/column and DELTA\_ROW/DELTA\_COLUMN is the NUMBER of indices to skip for each row/column index.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

#### See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_ROW** The first row of the matrix to be output

**DELTA\_ROW** The delta row increment to be used when outputting the first through to the last matrix row

**LAST\_ROW** The last row of the matrix to be output

**FIRST\_COLUMN** The first column of the matrix to be output

**DELTA\_COLUMN** The delta column increase to be used when outputting the first through to the last matrix column

**LAST\_COLUMN** The last column of the matrix to be output

**INDEX\_FORMAT\_TYPE** The format type to be used for the matrix name and indices

#### See also:

[INPUT\\_OUTPUT::MatrixNameIndexFormat](#), [INPUT\\_OUTPUT::MatrixNameIndexFormat](#)

**MATRIX\_NAME\_FORMAT** The format string to be used to format the matrix name

**ROW\_INDEX\_FORMAT** The format string to be used to format the row indices

**FIRST\_FORMAT** The format string to be used for the first line of output

**REPEAT\_FORMAT** The format type to be used for the second and subsequently repeated lines of output

**ERR** The error code

**ERROR** The error string

Definition at line 3922 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), [BASE\\_ROUTINES::WRITE\\_STR\(\)](#), [WRITE\\_STRING\\_MATRIX\\_NAME\\_AND\\_INDICES](#), and [WRITE\\_STRING\\_MATRIX\\_NAME\\_ONLY](#).

Here is the call graph for this function:

---

**6.25.2.60 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_C\_C**  
 (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*,  
 CHARACTER(LEN=\*),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in)  
*SECOND\_STRING*, CHARACTER(LEN=\*),intent(in) *SECOND\_VALUE*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*FIRST\_VALUE* The first value to be output

*SECOND\_STRING* The second string to be output

*SECOND\_VALUE* The second value to be output

*ERR* The error code

*ERROR* The error string

Definition at line 480 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.61 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_C\_DP**  
 (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*,  
 CHARACTER(LEN=\*),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in)  
*SECOND\_STRING*, REAL(DP),intent(in) *SECOND\_VALUE*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*FIRST\_VALUE* The first value to be output

*SECOND\_STRING* The second string to be output

*SECOND\_VALUE* The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 511 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.25.2.62 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_C\_INTG**  
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,`  
`CHARACTER(LEN=*),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)`  
`SECOND_STRING, INTEGER(INTG),intent(in) SECOND_VALUE,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`  
`*) [private]`

Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 543 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.25.2.63 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_C\_L**  
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,`  
`CHARACTER(LEN=*),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)`  
`SECOND_STRING, LOGICAL,intent(in) SECOND_VALUE,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`  
`*) [private]`

Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 575 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL\\_TO\\_VSTRING\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

```
6.25.2.64 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_SP
 (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
 CHARACTER(LEN=*),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
 SECOND_STRING, REAL(SP),intent(in) SECOND_VALUE,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *) [private]
```

Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 606 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.65 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_C\_VS**  
 (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*,  
 CHARACTER(LEN=\*),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in)  
*SECOND\_STRING*, TYPE(VARYING\_STRING),intent(in) *SECOND\_VALUE*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted varying string SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*FIRST\_VALUE* The first value to be output

*SECOND\_STRING* The second string to be output

*SECOND\_VALUE* The second value to be output

*ERR* The error code

*ERROR* The error string

Definition at line 638 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.66 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_DP\_C**  
 (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*,  
 REAL(DP),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in)  
*SECOND\_STRING*, CHARACTER(LEN=\*),intent(in) *SECOND\_VALUE*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*FIRST\_VALUE* The first value to be output

*SECOND\_STRING* The second string to be output

*SECOND\_VALUE* The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 669 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.67 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_DP
 (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
 REAL(DP),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in) SECOND_
 STRING, REAL(DP),intent(in) SECOND_VALUE, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 701 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.68 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_INTG
 (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
 REAL(DP),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
 SECOND_STRING, INTEGER(INTG),intent(in) SECOND_VALUE,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *) [private]
```

Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 737 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.69 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_DP\_L**  
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,  
REAL(DP),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in) SECOND_-  
STRING, LOGICAL,intent(in) SECOND_VALUE, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 773 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL\\_TO\\_VSTRING\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.70 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_DP\_SP**  
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,  
REAL(DP),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in) SECOND_-  
STRING, REAL(SP),intent(in) SECOND_VALUE, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output

stream specified by ID. Free format is used to format both values.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 808 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

```
6.25.2.71 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_VS
 (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
 REAL(DP),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
 SECOND_STRING, TYPE(VARYING_STRING),intent(in) SECOND_VALUE,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *) [private]
```

Writes the **FIRST\_STRING** followed by a formatted double precision **FIRST\_VALUE** and the **SECOND\_STRING** followed by a formatted single precision **SECOND\_VALUE** to the given output stream specified by ID. Free format is used to format both values.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 844 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.72 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_INTG\_C**  
 (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*,  
 INTEGER(INTG),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in)  
*SECOND\_STRING*, CHARACTER(LEN=\*),intent(in) *SECOND\_VALUE*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*FIRST\_VALUE* The first value to be output

*SECOND\_STRING* The second string to be output

*SECOND\_VALUE* The second value to be output

*ERR* The error code

*ERROR* The error string

Definition at line 876 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.73 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_INTG\_DP**  
 (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*,  
 INTEGER(INTG),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in)  
*SECOND\_STRING*, REAL(DP),intent(in) *SECOND\_VALUE*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*FIRST\_VALUE* The first value to be output

*SECOND\_STRING* The second string to be output

*SECOND\_VALUE* The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 908 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.74 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_INTG
 (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
 INTEGER(INTG),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
 SECOND_STRING, INTEGER(INTG),intent(in) SECOND_VALUE,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *) [private]
```

Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 944 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.75 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_L
 (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
 INTEGER(INTG),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
 SECOND_STRING, LOGICAL,intent(in) SECOND_VALUE,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *) [private]
```

Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 980 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL\\_TO\\_VSTRING\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

```
6.25.2.76 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_SP
 (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
 INTEGER(INTG),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
 SECOND_STRING, REAL(SP),intent(in) SECOND_VALUE,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *) [private]
```

Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 1015 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.77 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_INTG\_VS**  
 (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*,  
 INTEGER(INTG),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in)  
*SECOND\_STRING*, TYPE(VARYING\_STRING),intent(in) *SECOND\_VALUE*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*FIRST\_VALUE* The first value to be output

*SECOND\_STRING* The second string to be output

*SECOND\_VALUE* The second value to be output

*ERR* The error code

*ERROR* The error string

Definition at line 1051 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.78 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_L\_C**  
 (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*,  
 LOGICAL,intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in)  
*SECOND\_STRING*, CHARACTER(LEN=\*),intent(in) *SECOND\_VALUE*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*FIRST\_VALUE* The first value to be output

*SECOND\_STRING* The second string to be output

*SECOND\_VALUE* The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 1083 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `STRINGS::LOGICAL_TO_VSTRING()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.79 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_DP
 (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
 LOGICAL,intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in) SECOND_
 STRING, REAL(DP),intent(in) SECOND_VALUE, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the `FIRST_STRING` followed by a formatted logical `FIRST_VALUE` and the `SECOND_STRING` followed by a formatted double precision `SECOND_VALUE` to the given output stream specified by `ID`. Free format is used to format both values.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 1115 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `STRINGS::LOGICAL_TO_VSTRING()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.80 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_INTG
 (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
 LOGICAL,intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
 SECOND_STRING, INTEGER(INTG),intent(in) SECOND_VALUE,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *) [private]
```

Writes the `FIRST_STRING` followed by a formatted logical `FIRST_VALUE` and the `SECOND_STRING` followed by a formatted integer `SECOND_VALUE` to the given output stream specified by `ID`. Free format is used to format both values.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 1151 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL\\_TO\\_VSTRING\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.81 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_L\_L**  
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,  
 LOGICAL,intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in) SECOND_  
 STRING, LOGICAL,intent(in) SECOND_VALUE, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 1185 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL\\_TO\\_VSTRING\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.82 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_L\_SP**  
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,  
 LOGICAL,intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in) SECOND_  
 STRING, REAL(SP),intent(in) SECOND_VALUE, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified

by ID. Free format is used to format both values.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 1220 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL\\_TO\\_VSTRING\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

```
6.25.2.83 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_VS
 (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
 LOGICAL,intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
 SECOND_STRING, TYPE(VARYING_STRING),intent(in) SECOND_VALUE,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *) [private]
```

Writes the **FIRST\_STRING** followed by a formatted logical **FIRST\_VALUE** and the the **SECOND\_STRING** followed by a formatted single precision **SECOND\_VALUE** to the given output stream specified by *ID*. Free format is used to format both values.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 1254 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL\\_TO\\_VSTRING\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.84 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_SP\_C**  
 (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*,  
 REAL(SP),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in)  
*SECOND\_STRING*, CHARACTER(LEN=\*),intent(in) *SECOND\_VALUE*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Writes the *FIRST\_STRING* followed by a formatted single precision *FIRST\_VALUE* and the *SECOND\_STRING* followed by a formatted character *SECOND\_VALUE* to the given output stream specified by *ID*. Free format is used to format both values.

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 1286 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.85 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_SP\_DP**  
 (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*,  
 REAL(SP),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*,  
 REAL(DP),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Writes the *FIRST\_STRING* followed by a formatted single precision *FIRST\_VALUE* and the *SECOND\_STRING* followed by a formatted double precision *SECOND\_VALUE* to the given output stream specified by *ID*. Free format is used to format both values.

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

**ERROR** The error string

Definition at line 1318 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.86 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_INTG
 (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
 REAL(SP),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
 SECOND_STRING, INTEGER(INTG),intent(in) SECOND_VALUE,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *) [private]
```

Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 1354 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.87 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_L
 (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
 REAL(SP),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in) SECOND_
 -STRING, LOGICAL,intent(in) SECOND_VALUE, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 1390 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL\\_TO\\_VSTRING\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

**6.25.2.88 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_SP\_SP**  
**(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=\*),intent(in) FIRST\_STRING,**  
**REAL(SP),intent(in) FIRST\_VALUE, CHARACTER(LEN=\*),intent(in) SECOND\_-**  
**STRING, REAL(SP),intent(in) SECOND\_VALUE, INTEGER(INTG),intent(out) ERR,**  
**TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 1423 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.89 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_SP\_VS**  
 (**INTEGER(INTG),intent(in) ID**, **CHARACTER(LEN=\*)**,**intent(in) FIRST\_STRING**,  
**REAL(SP),intent(in) FIRST\_VALUE**, **CHARACTER(LEN=\*)**,**intent(in) SECOND\_STRING**, **TYPE(VARYING\_STRING)**,**intent(in) SECOND\_VALUE**,  
**INTEGER(INTG),intent(out) ERR**, **TYPE(VARYING\_STRING)**,**intent(out) ERROR**,  
 \*) [private]

Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 1459 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.90 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_VS\_C**  
 (**INTEGER(INTG),intent(in) ID**, **CHARACTER(LEN=\*)**,**intent(in) FIRST\_STRING**, **TYPE(VARYING\_STRING)**,**intent(in) FIRST\_VALUE**, **CHARACTER(LEN=\*)**,**intent(in) SECOND\_STRING**, **CHARACTER(LEN=\*)**,**intent(in) SECOND\_VALUE**, **INTEGER(INTG),intent(out) ERR**, **TYPE(VARYING\_STRING)**,**intent(out) ERROR**, \*) [private]

Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 1491 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.91 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_DP
 (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in)
 FIRST_STRING, TYPE(VARYING_STRING),intent(in) FIRST_VALUE,
 CHARACTER(LEN=*),intent(in) SECOND_STRING, REAL(DP),intent(in) SECOND_
 VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)
 ERROR, *) [private]
```

Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 1522 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.92 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_INTG
 (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in)
 FIRST_STRING, TYPE(VARYING_STRING),intent(in) FIRST_
 VALUE, CHARACTER(LEN=*),intent(in) SECOND_STRING,
 INTEGER(INTG),intent(in) SECOND_VALUE, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 1554 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

```
6.25.2.93 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_L
 (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in)
 FIRST_STRING, TYPE(VARYING_STRING),intent(in) FIRST_VALUE,
 CHARACTER(LEN=*),intent(in) SECOND_STRING, LOGICAL,intent(in) SECOND_
 VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)
 ERROR, *) [private]
```

Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

**Parameters:**

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 1586 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL\\_TO\\_VSTRING\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.94 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_VS\_SP**  
*(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=\*),intent(in)  
 FIRST\_STRING, TYPE(VARYING\_STRING),intent(in) FIRST\_VALUE,  
 CHARACTER(LEN=\*),intent(in) SECOND\_STRING, REAL(SP),intent(in) SECOND\_-  
 VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out)  
 ERROR, \*) [private]*

Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*FIRST\_VALUE* The first value to be output

*SECOND\_STRING* The second string to be output

*SECOND\_VALUE* The second value to be output

*ERR* The error code

*ERROR* The error string

Definition at line 1617 of file input\_output.f90.

References [BASE\\_ROUTINES::ERRORS\(\)](#), [BASE\\_ROUTINES::OP\\_STRING](#), and [BASE\\_ROUTINES::WRITE\\_STR\(\)](#).

Here is the call graph for this function:

---

**6.25.2.95 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_VS\_VS**  
*(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=\*),intent(in)  
 FIRST\_STRING, TYPE(VARYING\_STRING),intent(in) FIRST\_VALUE,  
 CHARACTER(LEN=\*),intent(in) SECOND\_STRING, TYPE(VARYING\_-  
 STRING),intent(in) SECOND\_VALUE, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]*

Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted varying string SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*FIRST\_VALUE* The first value to be output

*SECOND\_STRING* The second string to be output

*SECOND\_VALUE* The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 1649 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.25.2.96 subroutine INPUT\_OUTPUT::WRITE\_STRING\_VALUE\_C**  
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,`  
`CHARACTER(LEN=*),intent(in) VALUE, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**VALUE** The value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 273 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.25.2.97 subroutine INPUT\_OUTPUT::WRITE\_STRING\_VALUE\_DP**  
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in)`  
`FIRST_STRING, REAL(DP),intent(in) VALUE, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**VALUE** The value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 302 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.25.2.98 subroutine INPUT\_OUTPUT::WRITE\_STRING\_VALUE\_INTG**  
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,`  
`INTEGER(INTG),intent(in) VALUE, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**VALUE** The value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 332 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.25.2.99 subroutine INPUT\_OUTPUT::WRITE\_STRING\_VALUE\_L**  
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,`  
`LOGICAL,intent(in) VALUE, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**VALUE** The value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 392 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `STRINGS::LOGICAL_TO_VSTRING()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.100 subroutine INPUT_OUTPUT::WRITE_STRING_VALUE_LINTG
 (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*)intent(in) FIRST_STRING,
 INTEGER(LINTG),intent(in) VALUE, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, */ [private]
```

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**VALUE** The value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 362 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.101 subroutine INPUT_OUTPUT::WRITE_STRING_VALUE_SP
 (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*)intent(in)
 FIRST_STRING, REAL(SP),intent(in) VALUE, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, */ [private]
```

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**VALUE** The value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 421 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.102 subroutine INPUT_OUTPUT::WRITE_STRING_VALUE_VS
 (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*) intent(in) FIRST_STRING,
 TYPE(VARYING_STRING),intent(in) VALUE, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**VALUE** The value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 451 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.25.2.103 subroutine INPUT_OUTPUT::WRITE_STRING_VS (INTEGER(INTG),intent(in) ID,
 TYPE(VARYING_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the varying string STRING to the given output stream specified by ID.

#### Parameters:

**ID** The ID of the output stream to write to

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**STRING** The string to write

**ERR** The error code

**ERROR** The error string

Definition at line 247 of file input\_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

## 6.26 ISO\_VARYING\_STRING Namespace Reference

This module provides an iso\_varying\_string module, conformant to the API specified in ISO/IEC 1539-2:2000 (varying-length strings for Fortran 95).

### Classes

- struct `VARYING_STRING`
- interface `assignment(=)`
- interface `adjustr`
- interface `char`
- interface `iachar`
- interface `ichar`
- interface `index`
- interface `len`
- interface `len_trim`
- interface `lge`
- interface `lgt`
- interface `lle`
- interface `llt`
- interface `repeat`
- interface `scan`
- interface `trim`
- interface `verify`
- interface `var_str`
- interface `get`
- interface `put`
- interface `put_line`
- interface `extract`
- interface `insert`
- interface `remove`
- interface `replace`
- interface `split`

### Functions

- `integer len_(string)`
- subroutine `op_assign_CH_VS` (`var, exp`)
- subroutine `op_assign_VS_CH` (`var, exp`)
- type(varying\_string) `op_concat_VS_VS` (`string_a, string_b`)
- type(varying\_string) `op_concat_CH_VS` (`string_a, string_b`)
- type(varying\_string) `op_concat_VS_CH` (`string_a, string_b`)
- logical `op_eq_VS_VS` (`string_a, string_b`)
- logical `op_eq_CH_VS` (`string_a, string_b`)
- logical `op_eq_VS_CH` (`string_a, string_b`)
- logical `op_ne_VS_VS` (`string_a, string_b`)
- logical `op_ne_CH_VS` (`string_a, string_b`)
- logical `op_ne_VS_CH` (`string_a, string_b`)
- logical `op_lt_VS_VS` (`string_a, string_b`)

- `logical op_lt_CH_VS (string_a, string_b)`
- `logical op_lt_VS_CH (string_a, string_b)`
- `logical op_le_VS_VS (string_a, string_b)`
- `logical op_le_CH_VS (string_a, string_b)`
- `logical op_le_VS_CH (string_a, string_b)`
- `logical op_ge_VS_VS (string_a, string_b)`
- `logical op_ge_CH_VS (string_a, string_b)`
- `logical op_ge_VS_CH (string_a, string_b)`
- `logical op_gt_VS_VS (string_a, string_b)`
- `logical op_gt_CH_VS (string_a, string_b)`
- `logical op_gt_VS_CH (string_a, string_b)`
- `type(varying_string) adjustl_ (string)`
- `type(varying_string) adjustr_ (string)`
- `function char_auto (string)`
- `character(LEN=length) char_fixed (string, length)`
- `integer iachar_ (c)`
- `integer ichar_ (c)`
- `integer index_VS_VS (string, substring, back)`
- `integer index_CH_VS (string, substring, back)`
- `integer index_VS_CH (string, substring, back)`
- `integer len_trim_ (string)`
- `logical lge_VS_VS (string_a, string_b)`
- `logical lge_CH_VS (string_a, string_b)`
- `logical lge_VS_CH (string_a, string_b)`
- `logical lgt_VS_VS (string_a, string_b)`
- `logical lgt_CH_VS (string_a, string_b)`
- `logical lgt_VS_CH (string_a, string_b)`
- `logical lle_VS_VS (string_a, string_b)`
- `logical lle_CH_VS (string_a, string_b)`
- `logical lle_VS_CH (string_a, string_b)`
- `logical llt_VS_VS (string_a, string_b)`
- `logical llt_CH_VS (string_a, string_b)`
- `logical llt_VS_CH (string_a, string_b)`
- `type(varying_string) repeat_ (string, ncopies)`
- `integer scan_VS_VS (string, set, back)`
- `integer scan_CH_VS (string, set, back)`
- `integer scan_VS_CH (string, set, back)`
- `type(varying_string) trim_ (string)`
- `integer verify_VS_VS (string, set, back)`
- `integer verify_CH_VS (string, set, back)`
- `integer verify_VS_CH (string, set, back)`
- `type(varying_string) var_str_ (char)`
- `subroutine get_ (string, maxlen, iostat)`
- `subroutine get_unit (unit, string, maxlen, iostat)`
- `subroutine get_set_VS (string, set, separator, maxlen, iostat)`
- `subroutine get_set_CH (string, set, separator, maxlen, iostat)`
- `subroutine get_unit_set_VS (unit, string, set, separator, maxlen, iostat)`
- `subroutine get_unit_set_CH (unit, string, set, separator, maxlen, iostat)`
- `subroutine put_VS (string, iostat)`
- `subroutine put_CH (string, iostat)`

- subroutine `put_unit_VS` (unit, string, iostat)
- subroutine `put_unit_CH` (unit, string, iostat)
- subroutine `put_line_VS` (string, iostat)
- subroutine `put_line_CH` (string, iostat)
- subroutine `put_line_unit_VS` (unit, string, iostat)
- subroutine `put_line_unit_CH` (unit, string, iostat)
- type(varying\_string) `extract_VS` (string, start, finish)
- type(varying\_string) `extract_CH` (string, start, finish)
- type(varying\_string) `insert_VS_VS` (string, start, substring)
- type(varying\_string) `insert_CH_VS` (string, start, substring)
- type(varying\_string) `insert_VS_CH` (string, start, substring)
- type(varying\_string) `insert_CH_CH` (string, start, substring)
- TYPE(varying\_string) `remove_VS` (string, start, finish)
- type(varying\_string) `remove_CH` (string, start, finish)
- type(varying\_string) `replace_VS_VS_auto` (string, start, substring)
- type(varying\_string) `replace_CH_VS_auto` (string, start, substring)
- type(varying\_string) `replace_VS_CH_auto` (string, start, substring)
- type(varying\_string) `replace_CH_CH_auto` (string, start, substring)
- type(varying\_string) `replace_VS_VS_fixed` (string, start, finish, substring)
- type(varying\_string) `replace_CH_VS_fixed` (string, start, finish, substring)
- type(varying\_string) `replace_VS_CH_fixed` (string, start, finish, substring)
- type(varying\_string) `replace_CH_CH_fixed` (string, start, finish, substring)
- type(varying\_string) `replace_VS_VS_VS_target` (string, target, substring, every, back)
- type(varying\_string) `replace_CH_VS_VS_target` (string, target, substring, every, back)
- type(varying\_string) `replace_VS_CH_VS_target` (string, target, substring, every, back)
- type(varying\_string) `replace_CH_CH_VS_target` (string, target, substring, every, back)
- type(varying\_string) `replace_VS_VS_CH_target` (string, target, substring, every, back)
- type(varying\_string) `replace_CH_VS_CH_target` (string, target, substring, every, back)
- type(varying\_string) `replace_VS_CH_CH_target` (string, target, substring, every, back)
- type(varying\_string) `replace_CH_CH_CH_target` (string, target, substring, every, back)
- subroutine `split_VS` (string, word, set, separator, back)
- subroutine `split_CH` (string, word, set, separator, back)

## Variables

- `integer`, parameter `GET_BUFFER_LEN` = 256

### 6.26.1 Detailed Description

This module provides an iso\_varying\_string module, conformant to the API specified in ISO/IEC 1539-2:2000 (varying-length strings for Fortran 95).

### 6.26.2 Function Documentation

#### 6.26.2.1 type(varying\_string) ISO\_VARYING\_STRING::adjustl\_ (type(varying\_string),intent(in) string) [private]

Definition at line 858 of file iso\_varying\_string.f90.

---

**6.26.2.2 type(varying\_string) ISO\_VARYING\_STRING::adjustr\_ (type(varying\_string),intent(in) string) [private]**

Definition at line 875 of file iso\_varying\_string.f90.

**6.26.2.3 function ISO\_VARYING\_STRING::char\_auto (type(varying\_string),intent(in) string) [private]**

Definition at line 892 of file iso\_varying\_string.f90.

**6.26.2.4 character(LEN=length) ISO\_VARYING\_STRING::char\_fixed  
(type(varying\_string),intent(in) string, integer,intent(in) length) [private]**

Definition at line 914 of file iso\_varying\_string.f90.

**6.26.2.5 type(varying\_string) ISO\_VARYING\_STRING::extract\_CH  
(character(LEN=\*)intent(in) string, integer,intent(in),optional start,  
integer,intent(in),optional finish) [private]**

Definition at line 1901 of file iso\_varying\_string.f90.

**6.26.2.6 type(varying\_string) ISO\_VARYING\_STRING::extract\_VS (type(varying\_string),intent(in) string, integer,intent(in),optional start, integer,intent(in),optional finish) [private]**

Definition at line 1882 of file iso\_varying\_string.f90.

**6.26.2.7 subroutine ISO\_VARYING\_STRING::get\_ (type(varying\_string),intent(out) string, integer,intent(in),optional maxlen, integer,intent(out),optional iostat) [private]**

Definition at line 1455 of file iso\_varying\_string.f90.

References GET\_BUFFER\_LEN.

**6.26.2.8 subroutine ISO\_VARYING\_STRING::get\_set\_CH (type(varying\_string),intent(out) string, character(LEN=\*)intent(in) set, type(varying\_string),intent(out),optional separator, integer,intent(in),optional maxlen, integer,intent(out),optional iostat) [private]**

Definition at line 1583 of file iso\_varying\_string.f90.

**6.26.2.9 subroutine ISO\_VARYING\_STRING::get\_set\_VS (type(varying\_string),intent(out) string, type(varying\_string),intent(in) set, type(varying\_string),intent(out),optional separator, integer,intent(in),optional maxlen, integer,intent(out),optional iostat) [private]**

Definition at line 1562 of file iso\_varying\_string.f90.

**6.26.2.10 subroutine ISO\_VARYING\_STRING::get\_unit (integer,intent(in) *unit*, type(varying\_string),intent(out) *string*, integer,intent(in),optional *maxlen*, integer,intent(out),optional *iostat*) [private]**

Definition at line 1508 of file iso\_varying\_string.f90.

References GET\_BUFFER\_LEN.

**6.26.2.11 subroutine ISO\_VARYING\_STRING::get\_unit\_set\_CH (integer,intent(in) *unit*, type(varying\_string),intent(out) *string*, character(LEN=\*),intent(in) *set*, type(varying\_string),intent(out),optional *separator*, integer,intent(in),optional *maxlen*, integer,intent(out),optional *iostat*) [private]**

Definition at line 1663 of file iso\_varying\_string.f90.

**6.26.2.12 subroutine ISO\_VARYING\_STRING::get\_unit\_set\_VS (integer,intent(in) *unit*, type(varying\_string),intent(out) *string*, type(varying\_string),intent(in) *set*, type(varying\_string),intent(out),optional *separator*, integer,intent(in),optional *maxlen*, integer,intent(out),optional *iostat*) [private]**

Definition at line 1641 of file iso\_varying\_string.f90.

**6.26.2.13 integer ISO\_VARYING\_STRING::iachar\_ (type(varying\_string),intent(in) *c*) [private]**

Definition at line 933 of file iso\_varying\_string.f90.

**6.26.2.14 integer ISO\_VARYING\_STRING::ichar\_ (type(varying\_string),intent(in) *c*) [private]**

Definition at line 951 of file iso\_varying\_string.f90.

**6.26.2.15 integer ISO\_VARYING\_STRING::index\_CH\_VS (character(LEN=\*),intent(in) *string*, type(varying\_string),intent(in) *substring*, logical,intent(in),optional *back*) [private]**

Definition at line 989 of file iso\_varying\_string.f90.

**6.26.2.16 integer ISO\_VARYING\_STRING::index\_VS\_CH (type(varying\_string),intent(in) *string*, character(LEN=\*),intent(in) *substring*, logical,intent(in),optional *back*) [private]**

Definition at line 1009 of file iso\_varying\_string.f90.

**6.26.2.17 integer ISO\_VARYING\_STRING::index\_VS\_VS (type(varying\_string),intent(in) *string*, type(varying\_string),intent(in) *substring*, logical,intent(in),optional *back*) [private]**

Definition at line 969 of file iso\_varying\_string.f90.

---

**6.26.2.18 type(varying\_string) ISO\_VARYING\_STRING::insert\_CH\_CH  
(character(LEN=\*),intent(in) *string*, integer,intent(in) *start*,  
character(LEN=\*),intent(in) *substring*) [private]**

Definition at line 1992 of file iso\_varying\_string.f90.

**6.26.2.19 type(varying\_string) ISO\_VARYING\_STRING::insert\_CH\_VS  
(character(LEN=\*),intent(in) *string*, integer,intent(in) *start*,  
type(varying\_string),intent(in) *substring*) [private]**

Definition at line 1954 of file iso\_varying\_string.f90.

**6.26.2.20 type(varying\_string) ISO\_VARYING\_STRING::insert\_VS\_CH (type(varying\_-  
string),intent(in) *string*, integer,intent(in) *start*, character(LEN=\*),intent(in) *substring*)  
[private]**

Definition at line 1973 of file iso\_varying\_string.f90.

**6.26.2.21 type(varying\_string) ISO\_VARYING\_STRING::insert\_VS\_VS (type(varying\_-  
string),intent(in) *string*, integer,intent(in) *start*, type(varying\_string),intent(in)  
*substring*) [private]**

Definition at line 1935 of file iso\_varying\_string.f90.

**6.26.2.22 integer ISO\_VARYING\_STRING::len\_ (type(varying\_string),intent(in) *string*)  
[private]**

Definition at line 398 of file iso\_varying\_string.f90.

**6.26.2.23 integer ISO\_VARYING\_STRING::len\_trim\_ (type(varying\_string),intent(in) *string*)  
[private]**

Definition at line 1029 of file iso\_varying\_string.f90.

**6.26.2.24 logical ISO\_VARYING\_STRING::lge\_CH\_VS (character(LEN=\*),intent(in) *string\_a*,  
type(varying\_string),intent(in) *string\_b*) [private]**

Definition at line 1068 of file iso\_varying\_string.f90.

**6.26.2.25 logical ISO\_VARYING\_STRING::lge\_VS\_CH (type(varying\_string),intent(in) *string\_a*,  
character(LEN=\*),intent(in) *string\_b*) [private]**

Definition at line 1087 of file iso\_varying\_string.f90.

**6.26.2.26 logical ISO\_VARYING\_STRING::lge\_VS\_VS (type(varying\_string),intent(in) *string\_a*,  
type(varying\_string),intent(in) *string\_b*) [private]**

Definition at line 1050 of file iso\_varying\_string.f90.

**6.26.2.27 logical ISO\_VARYING\_STRING::lgt\_CH\_VS (character(LEN=\*),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*) [private]**

Definition at line 1124 of file iso\_varying\_string.f90.

**6.26.2.28 logical ISO\_VARYING\_STRING::lgt\_VS\_CH (type(varying\_string),intent(in) *string\_a*, character(LEN=\*),intent(in) *string\_b*) [private]**

Definition at line 1143 of file iso\_varying\_string.f90.

**6.26.2.29 logical ISO\_VARYING\_STRING::lgt\_VS\_VS (type(varying\_string),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*) [private]**

Definition at line 1106 of file iso\_varying\_string.f90.

**6.26.2.30 logical ISO\_VARYING\_STRING::lle\_CH\_VS (character(LEN=\*),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*) [private]**

Definition at line 1180 of file iso\_varying\_string.f90.

**6.26.2.31 logical ISO\_VARYING\_STRING::lle\_VS\_CH (type(varying\_string),intent(in) *string\_a*, character(LEN=\*),intent(in) *string\_b*) [private]**

Definition at line 1199 of file iso\_varying\_string.f90.

**6.26.2.32 logical ISO\_VARYING\_STRING::lle\_VS\_VS (type(varying\_string),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*) [private]**

Definition at line 1162 of file iso\_varying\_string.f90.

**6.26.2.33 logical ISO\_VARYING\_STRING::llt\_CH\_VS (character(LEN=\*),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*) [private]**

Definition at line 1236 of file iso\_varying\_string.f90.

**6.26.2.34 logical ISO\_VARYING\_STRING::llt\_VS\_CH (type(varying\_string),intent(in) *string\_a*, character(LEN=\*),intent(in) *string\_b*) [private]**

Definition at line 1255 of file iso\_varying\_string.f90.

**6.26.2.35 logical ISO\_VARYING\_STRING::llt\_VS\_VS (type(varying\_string),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*) [private]**

Definition at line 1218 of file iso\_varying\_string.f90.

---

**6.26.2.36 subroutine ISO\_VARYING\_STRING::op\_assign\_CH\_VS**  
`(character(LEN=*)&,intent(out) var, type(varying_string),intent(in) exp) [private]`

Definition at line 419 of file iso\_varying\_string.f90.

**6.26.2.37 subroutine ISO\_VARYING\_STRING::op\_assign\_VS\_CH**  
`(type(varying_string),intent(out) var, character(LEN=*)&,intent(in) exp) [private]`

Definition at line 436 of file iso\_varying\_string.f90.

**6.26.2.38 type(varying\_string) ISO\_VARYING\_STRING::op\_concat\_CH\_VS**  
`(character(LEN=*)&,intent(in) string_a, type(varying_string),intent(in) string_b)`  
`[private]`

Definition at line 484 of file iso\_varying\_string.f90.

References op\_concat\_VS\_VS().

Here is the call graph for this function:

**6.26.2.39 type(varying\_string) ISO\_VARYING\_STRING::op\_concat\_VS\_CH**  
`(type(varying_string),intent(in) string_a, character(LEN=*)&,intent(in) string_b)`  
`[private]`

Definition at line 503 of file iso\_varying\_string.f90.

References op\_concat\_VS\_VS().

Here is the call graph for this function:

**6.26.2.40 type(varying\_string) ISO\_VARYING\_STRING::op\_concat\_VS\_VS**  
`(type(varying_string),intent(in) string_a, type(varying_string),intent(in) string_b)`  
`[private]`

Definition at line 453 of file iso\_varying\_string.f90.

Referenced by op\_concat\_CH\_VS(), and op\_concat\_VS\_CH().

Here is the caller graph for this function:

**6.26.2.41 logical ISO\_VARYING\_STRING::op\_eq\_CH\_VS** `(character(LEN=*)&,intent(in) string_a, type(varying_string),intent(in) string_b)` [private]

Definition at line 540 of file iso\_varying\_string.f90.

**6.26.2.42 logical ISO\_VARYING\_STRING::op\_eq\_VS\_CH** `(type(varying_string),intent(in) string_a, character(LEN=*)&,intent(in) string_b)` [private]

Definition at line 559 of file iso\_varying\_string.f90.

**6.26.2.43 logical ISO\_VARYING\_STRING::op\_eq\_VS\_VS (type(varying\_string),intent(in) string\_a, type(varying\_string),intent(in) string\_b) [private]**

Definition at line 522 of file iso\_varying\_string.f90.

**6.26.2.44 logical ISO\_VARYING\_STRING::op\_ge\_CH\_VS (character(LEN=\*),intent(in) string\_a, type(varying\_string),intent(in) string\_b) [private]**

Definition at line 764 of file iso\_varying\_string.f90.

**6.26.2.45 logical ISO\_VARYING\_STRING::op\_ge\_VS\_CH (type(varying\_string),intent(in) string\_a, character(LEN=\*),intent(in) string\_b) [private]**

Definition at line 783 of file iso\_varying\_string.f90.

**6.26.2.46 logical ISO\_VARYING\_STRING::op\_ge\_VS\_VS (type(varying\_string),intent(in) string\_a, type(varying\_string),intent(in) string\_b) [private]**

Definition at line 746 of file iso\_varying\_string.f90.

**6.26.2.47 logical ISO\_VARYING\_STRING::op\_gt\_CH\_VS (character(LEN=\*),intent(in) string\_a, type(varying\_string),intent(in) string\_b) [private]**

Definition at line 820 of file iso\_varying\_string.f90.

**6.26.2.48 logical ISO\_VARYING\_STRING::op\_gt\_VS\_CH (type(varying\_string),intent(in) string\_a, character(LEN=\*),intent(in) string\_b) [private]**

Definition at line 839 of file iso\_varying\_string.f90.

**6.26.2.49 logical ISO\_VARYING\_STRING::op\_gt\_VS\_VS (type(varying\_string),intent(in) string\_a, type(varying\_string),intent(in) string\_b) [private]**

Definition at line 802 of file iso\_varying\_string.f90.

**6.26.2.50 logical ISO\_VARYING\_STRING::op\_le\_CH\_VS (character(LEN=\*),intent(in) string\_a, type(varying\_string),intent(in) string\_b) [private]**

Definition at line 708 of file iso\_varying\_string.f90.

**6.26.2.51 logical ISO\_VARYING\_STRING::op\_le\_VS\_CH (type(varying\_string),intent(in) string\_a, character(LEN=\*),intent(in) string\_b) [private]**

Definition at line 727 of file iso\_varying\_string.f90.

---

**6.26.2.52 logical ISO\_VARYING\_STRING::op\_le\_VS\_VS (type(varying\_string),intent(in) string\_a, type(varying\_string),intent(in) string\_b) [private]**

Definition at line 690 of file iso\_varying\_string.f90.

**6.26.2.53 logical ISO\_VARYING\_STRING::op\_lt\_CH\_VS (character(LEN=\*),intent(in) string\_a, type(varying\_string),intent(in) string\_b) [private]**

Definition at line 652 of file iso\_varying\_string.f90.

**6.26.2.54 logical ISO\_VARYING\_STRING::op\_lt\_VS\_CH (type(varying\_string),intent(in) string\_a, character(LEN=\*),intent(in) string\_b) [private]**

Definition at line 671 of file iso\_varying\_string.f90.

**6.26.2.55 logical ISO\_VARYING\_STRING::op\_lt\_VS\_VS (type(varying\_string),intent(in) string\_a, type(varying\_string),intent(in) string\_b) [private]**

Definition at line 634 of file iso\_varying\_string.f90.

**6.26.2.56 logical ISO\_VARYING\_STRING::op\_ne\_CH\_VS (character(LEN=\*),intent(in) string\_a, type(varying\_string),intent(in) string\_b) [private]**

Definition at line 596 of file iso\_varying\_string.f90.

**6.26.2.57 logical ISO\_VARYING\_STRING::op\_ne\_VS\_CH (type(varying\_string),intent(in) string\_a, character(LEN=\*),intent(in) string\_b) [private]**

Definition at line 615 of file iso\_varying\_string.f90.

**6.26.2.58 logical ISO\_VARYING\_STRING::op\_ne\_VS\_VS (type(varying\_string),intent(in) string\_a, type(varying\_string),intent(in) string\_b) [private]**

Definition at line 578 of file iso\_varying\_string.f90.

**6.26.2.59 subroutine ISO\_VARYING\_STRING::put\_CH (character(LEN=\*),intent(in) string, integer,intent(out),optional iostat) [private]**

Definition at line 1738 of file iso\_varying\_string.f90.

**6.26.2.60 subroutine ISO\_VARYING\_STRING::put\_line\_CH (character(LEN=\*),intent(in) string, integer,intent(out),optional iostat) [private]**

Definition at line 1818 of file iso\_varying\_string.f90.

**6.26.2.61 subroutine ISO\_VARYING\_STRING::put\_line\_unit\_CH (integer,intent(in) *unit*, character(LEN=\*),intent(in) *string*, integer,intent(out),optional *iostat*) [private]**

Definition at line 1859 of file iso\_varying\_string.f90.

**6.26.2.62 subroutine ISO\_VARYING\_STRING::put\_line\_unit\_VS (integer,intent(in) *unit*, type(varying\_string),intent(in) *string*, integer,intent(out),optional *iostat*) [private]**

Definition at line 1840 of file iso\_varying\_string.f90.

**6.26.2.63 subroutine ISO\_VARYING\_STRING::put\_line\_VS (type(varying\_string),intent(in) *string*, integer,intent(out),optional *iostat*) [private]**

Definition at line 1800 of file iso\_varying\_string.f90.

**6.26.2.64 subroutine ISO\_VARYING\_STRING::put\_unit\_CH (integer,intent(in) *unit*, character(LEN=\*),intent(in) *string*, integer,intent(out),optional *iostat*) [private]**

Definition at line 1777 of file iso\_varying\_string.f90.

**6.26.2.65 subroutine ISO\_VARYING\_STRING::put\_unit\_VS (integer,intent(in) *unit*, type(varying\_string),intent(in) *string*, integer,intent(out),optional *iostat*) [private]**

Definition at line 1758 of file iso\_varying\_string.f90.

**6.26.2.66 subroutine ISO\_VARYING\_STRING::put\_VS (type(varying\_string),intent(in) *string*, integer,intent(out),optional *iostat*) [private]**

Definition at line 1722 of file iso\_varying\_string.f90.

**6.26.2.67 type(varying\_string) ISO\_VARYING\_STRING::remove\_CH  
(character(LEN=\*),intent(in) *string*, integer,intent(in),optional *start*,  
integer,intent(in),optional *finish*) [private]**

Definition at line 2035 of file iso\_varying\_string.f90.

**6.26.2.68 TYPE(varying\_string) ISO\_VARYING\_STRING::remove\_VS (type(varying\_string),intent(in) *string*, integer,intent(in),optional *start*, integer,intent(in),optional *finish*) [private]**

Definition at line 2016 of file iso\_varying\_string.f90.

**6.26.2.69 type(varying\_string) ISO\_VARYING\_STRING::repeat\_ (type(varying\_string),intent(in) *string*, integer,intent(in) *ncopies*) [private]**

Definition at line 1274 of file iso\_varying\_string.f90.

---

**6.26.2.70 type(varying\_string) ISO\_VARYING\_STRING::replace\_CH\_CH\_auto**  
 (character(LEN=\*),intent(in) *string*, integer,intent(in) *start*,  
 character(LEN=\*),intent(in) *substring*) [private]

Definition at line 2133 of file iso\_varying\_string.f90.

**6.26.2.71 type(varying\_string) ISO\_VARYING\_STRING::replace\_CH\_CH\_CH\_target**  
 (character(LEN=\*),intent(in) *string*, character(LEN=\*),intent(in) *target*,  
 character(LEN=\*),intent(in) *substring*, logical,intent(in),optional *every*,  
 logical,intent(in),optional *back*) [private]

Definition at line 2410 of file iso\_varying\_string.f90.

**6.26.2.72 type(varying\_string) ISO\_VARYING\_STRING::replace\_CH\_CH\_fixed**  
 (character(LEN=\*),intent(in) *string*, integer,intent(in) *start*, integer,intent(in) *finish*,  
 character(LEN=\*),intent(in) *substring*) [private]

Definition at line 2218 of file iso\_varying\_string.f90.

**6.26.2.73 type(varying\_string) ISO\_VARYING\_STRING::replace\_CH\_CH\_VS\_target**  
 (character(LEN=\*),intent(in) *string*, character(LEN=\*),intent(in) *target*,  
 type(varying\_string),intent(in) *substring*, logical,intent(in),optional *every*,  
 logical,intent(in),optional *back*) [private]

Definition at line 2318 of file iso\_varying\_string.f90.

**6.26.2.74 type(varying\_string) ISO\_VARYING\_STRING::replace\_CH\_VS\_auto**  
 (character(LEN=\*),intent(in) *string*, integer,intent(in) *start*,  
 type(varying\_string),intent(in) *substring*) [private]

Definition at line 2093 of file iso\_varying\_string.f90.

**6.26.2.75 type(varying\_string) ISO\_VARYING\_STRING::replace\_CH\_VS\_CH\_target**  
 (character(LEN=\*),intent(in) *string*, type(varying\_string),intent(in) *target*,  
 character(LEN=\*),intent(in) *substring*, logical,intent(in),optional *every*,  
 logical,intent(in),optional *back*) [private]

Definition at line 2364 of file iso\_varying\_string.f90.

**6.26.2.76 type(varying\_string) ISO\_VARYING\_STRING::replace\_CH\_VS\_fixed**  
 (character(LEN=\*),intent(in) *string*, integer,intent(in) *start*, integer,intent(in) *finish*,  
 type(varying\_string),intent(in) *substring*) [private]

Definition at line 2176 of file iso\_varying\_string.f90.

**6.26.2.77 type(varying\_string) ISO\_VARYING\_STRING::replace\_CH\_VS\_VS\_target  
(character(LEN=\*),intent(in) *string*, type(varying\_string),intent(in) *target*,  
type(varying\_string),intent(in) *substring*, logical,intent(in),optional *every*,  
logical,intent(in),optional *back*) [private]**

Definition at line 2272 of file iso\_varying\_string.f90.

**6.26.2.78 type(varying\_string) ISO\_VARYING\_STRING::replace\_VS\_CH\_auto (type(varying\_-  
string),intent(in) *string*, integer,intent(in) *start*, character(LEN=\*),intent(in) *substring*)  
[private]**

Definition at line 2113 of file iso\_varying\_string.f90.

**6.26.2.79 type(varying\_string) ISO\_VARYING\_STRING::replace\_VS\_CH\_CH\_target  
(type(varying\_string),intent(in) *string*, character(LEN=\*),intent(in) *target*,  
character(LEN=\*),intent(in) *substring*, logical,intent(in),optional *every*,  
logical,intent(in),optional *back*) [private]**

Definition at line 2387 of file iso\_varying\_string.f90.

**6.26.2.80 type(varying\_string) ISO\_VARYING\_STRING::replace\_VS\_CH\_fixed  
(type(varying\_string),intent(in) *string*, integer,intent(in) *start*, integer,intent(in) *finish*,  
character(LEN=\*),intent(in) *substring*) [private]**

Definition at line 2197 of file iso\_varying\_string.f90.

**6.26.2.81 type(varying\_string) ISO\_VARYING\_STRING::replace\_VS\_CH\_VS\_target  
(type(varying\_string),intent(in) *string*, character(LEN=\*),intent(in) *target*,  
type(varying\_string),intent(in) *substring*, logical,intent(in),optional *every*,  
logical,intent(in),optional *back*) [private]**

Definition at line 2295 of file iso\_varying\_string.f90.

**6.26.2.82 type(varying\_string) ISO\_VARYING\_STRING::replace\_VS\_VS\_auto (type(varying\_-  
string),intent(in) *string*, integer,intent(in) *start*, type(varying\_string),intent(in)  
*substring*) [private]**

Definition at line 2073 of file iso\_varying\_string.f90.

**6.26.2.83 type(varying\_string) ISO\_VARYING\_STRING::replace\_VS\_VS\_CH\_target  
(type(varying\_string),intent(in) *string*, type(varying\_string),intent(in) *target*,  
character(LEN=\*),intent(in) *substring*, logical,intent(in),optional *every*,  
logical,intent(in),optional *back*) [private]**

Definition at line 2341 of file iso\_varying\_string.f90.

---

**6.26.2.84** `type(varying_string) ISO_VARYING_STRING::replace_VS_VS_fixed  
(type(varying_string),intent(in) string, integer,intent(in) start, integer,intent(in) finish,  
type(varying_string),intent(in) substring) [private]`

Definition at line 2153 of file iso\_varying\_string.f90.

**6.26.2.85** `type(varying_string) ISO_VARYING_STRING::replace_VS_VS_VS_target  
(type(varying_string),intent(in) string, type(varying_string),intent(in) target,  
type(varying_string),intent(in) substring, logical,intent(in),optional every,  
logical,intent(in),optional back) [private]`

Definition at line 2249 of file iso\_varying\_string.f90.

**6.26.2.86** `integer ISO_VARYING_STRING::scan_CH_VS (character(LEN=*)intent(in) string,  
type(varying_string),intent(in) set, logical,intent(in),optional back) [private]`

Definition at line 1312 of file iso\_varying\_string.f90.

**6.26.2.87** `integer ISO_VARYING_STRING::scan_VS_CH (type(varying_string),intent(in) string,  
character(LEN=*)intent(in) set, logical,intent(in),optional back) [private]`

Definition at line 1332 of file iso\_varying\_string.f90.

**6.26.2.88** `integer ISO_VARYING_STRING::scan_VS_VS (type(varying_string),intent(in) string,  
type(varying_string),intent(in) set, logical,intent(in),optional back) [private]`

Definition at line 1292 of file iso\_varying\_string.f90.

**6.26.2.89** `subroutine ISO_VARYING_STRING::split_CH (type(varying_string),intent(inout)  
string, type(varying_string),intent(out) word, character(LEN=*)intent(in) set,  
type(varying_string),intent(out),optional separator, logical,intent(in),optional back)  
[private]`

Definition at line 2514 of file iso\_varying\_string.f90.

Referenced by split\_VS().

Here is the caller graph for this function:

**6.26.2.90** `subroutine ISO_VARYING_STRING::split_VS (type(varying_string),intent(inout)  
string, type(varying_string),intent(out) word, type(varying_string),intent(in) set,  
type(varying_string),intent(out),optional separator, logical,intent(in),optional back)  
[private]`

Definition at line 2494 of file iso\_varying\_string.f90.

References split\_CH().

Here is the call graph for this function:

**6.26.2.91 type(varying\_string) ISO\_VARYING\_STRING::trim\_ (type(varying\_string),intent(in) string) [private]**

Definition at line 1352 of file iso\_varying\_string.f90.

**6.26.2.92 type(varying\_string) ISO\_VARYING\_STRING::var\_str\_ (character(LEN=\*)intent(in) char) [private]**

Definition at line 1429 of file iso\_varying\_string.f90.

**6.26.2.93 integer ISO\_VARYING\_STRING::verify\_CH\_VS (character(LEN=\*)intent(in) string, type(varying\_string),intent(in) set, logical,intent(in),optional back) [private]**

Definition at line 1389 of file iso\_varying\_string.f90.

**6.26.2.94 integer ISO\_VARYING\_STRING::verify\_VS\_CH (type(varying\_string),intent(in) string, character(LEN=\*)intent(in) set, logical,intent(in),optional back) [private]**

Definition at line 1409 of file iso\_varying\_string.f90.

**6.26.2.95 integer ISO\_VARYING\_STRING::verify\_VS\_VS (type(varying\_string),intent(in) string, type(varying\_string),intent(in) set, logical,intent(in),optional back) [private]**

Definition at line 1369 of file iso\_varying\_string.f90.

### 6.26.3 Variable Documentation

**6.26.3.1 integer,parameter ISO\_VARYING\_STRING::GET\_BUFFER\_LEN = 256**

Definition at line 54 of file iso\_varying\_string.f90.

Referenced by get\_(), and get\_unit().

## 6.27 KINDS Namespace Reference

This module contains all kind definitions.

### Variables

- INTEGER, parameter **INTG** = SELECTED\_INT\_KIND(9)  
*Standard integer kind.*
- INTEGER, parameter **SINTG** = SELECTED\_INT\_KIND(4)  
*Short integer kind.*
- INTEGER, parameter **LINTG** = SELECTED\_INT\_KIND(18)  
*Long integer kind.*
- INTEGER, parameter **PTR** = **INTG**  
*Pointer integer kind.*
- INTEGER, parameter **SP** = SELECTED\_REAL\_KIND(6)  
*Single precision real kind.*
- INTEGER, parameter **DP** = SELECTED\_REAL\_KIND(15)  
*Double precision real kind.*
- INTEGER, parameter **QP** = SELECTED\_REAL\_KIND(30)  
*Quadruple precision real kind.*
- INTEGER, parameter **SPC** = KIND((1.0\_SP)
- INTEGER, parameter **\_SP**  
*Single precision complex kind.*
- INTEGER, parameter **DPC** = KIND((1.0\_DP)
- INTEGER, parameter **\_DP**  
*Double precision complex kind.*

### 6.27.1 Detailed Description

This module contains all kind definitions.

## 6.28 LAPACK Namespace Reference

This module contains the [interface](#) descriptions to the [LAPACK](#) routines.

### Classes

- interface [interface](#)

#### 6.28.1 Detailed Description

This module contains the [interface](#) descriptions to the [LAPACK](#) routines.

## 6.29 LAPLACE\_EQUATIONS\_ROUTINES Namespace Reference

This module handles all Laplace equations routines.

### Functions

- subroutine [LAPLACE\\_EQUIPMENT\\_ANALYTIC\\_CALCULATE](#) (FIELD, EQUATION\_NUMBER, GEOMETRIC\_PARAMETERS, VARIABLE\_NUMBER, COMPONENT\_NUMBER, NODE\_NUMBER, DERIV\_NUMBER, VALUE, ERR, ERROR,\*)

*Calculates the element stiffness matrices and RHS for a Laplace equation finite element equations set.*

- subroutine [LAPLACE\\_EQUIPMENT\\_INTE](#) (FIELD, NODE\_NUMBER, COMPONENT\_NUMBER, VARIABLE\_NUMBER, NUMBER\_OF\_DIMENSIONS,&INTERPOLATED\_POINT, ERR, ERROR,\*)

*Gets the intepolated points for the particular node. Assume the node allocation for an element is fixed.*

- subroutine [LAPLACE\\_EQUIPMENT\\_FIXED\\_CONDITIONS\\_](#) (EQUATIONS\_SET, SET\_TYPE, DERIVATIVE\_NUMBER, NODE\_NUMBER, COMPONENT\_NUMBER,&VARIABLE\_NUMBER, CONDITION, VALUE, ERR, ERROR,\*)

*Sets a fixed condition for the equation set on the specified node. This is more generic method, move to other place.*

- subroutine [LAPLACE\\_EQUIPMENT\\_ANALYTIC\\_PARAMETER\\_SET\\_UPDATE](#) (EQUATIONS\_SET, GEOMETRIC\_PARAMETERS, NODE\_TYPE, ERR, ERROR,\*)

*Calculates the element stiffness matrices and RHS for a Laplace equation finite element equations set.*

- subroutine [LAPLACE\\_EQUIPMENTFINITE\\_ELEMENT\\_CALCULATE](#) (EQUATIONS\_SET, ELEMENT\_NUMBER, ERR, ERROR,\*)

*Calculates the element stiffness matrices and RHS for a Laplace equation finite element equations set.*

- subroutine [LAPLACE\\_EQUIPMENT\\_EQUATIONS\\_SET\\_SETUP](#) (EQUATIONS\_SET, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)

*Sets up the Laplace equation type of a classical field equations set class.*

- subroutine [LAPLACE\\_EQUIPMENT\\_EQUATIONS\\_SET\\_SUBTYPE\\_SET](#) (EQUATIONS\_SET, EQUATIONS\_SET\_SUBTYPE, ERR, ERROR,\*)

*Sets/changes the equation subtype for a Laplace equation type of a classical field equations set class.*

- subroutine [LAPLACE\\_EQUIPMENT\\_EQUATIONS\\_SET\\_STANDARD\\_SETUP](#) (EQUATIONS\_SET, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)

*Sets up the standard Laplace equation.*

- subroutine [LAPLACE\\_EQUIPMENT\\_PROBLEM\\_SETUP](#) (PROBLEM, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)

*Sets up the Laplace solution.*

- subroutine [LAPLACE\\_EQUIPMENT\\_PROBLEM\\_SUBTYPE\\_SET](#) (PROBLEM, PROBLEM\_SUBTYPE, ERR, ERROR,\*)

*Sets/changes the problem subtype for a Laplace equation type .*

- subroutine [LAPLACE\\_EQUATION\\_PROBLEM\\_STANDARD\\_SETUP](#) (PROBLEM, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)

*Sets up the standard Laplace equations solution.*

## Variables

- INTEGER(INTG), parameter [LAPLACE\\_EQUATION\\_TWO\\_DIM\\_1](#) = 1  
 $u=x^{**2}+2*x*y-y^{**2}$
- INTEGER(INTG), parameter [LAPLACE\\_EQUATION\\_TWO\\_DIM\\_2](#) = 2  
 $u=\cos(x)\cosh(y)$
- INTEGER(INTG), parameter [LAPLACE\\_EQUATION\\_THREE\\_DIM\\_1](#) = 3  
 $u=x^{**2}-2*y^{**2}+z^{**2}$
- INTEGER(INTG), parameter [LAPLACE\\_EQUATION\\_THREE\\_DIM\\_2](#) = 4  
 $u=\cos(x)*\cosh(y)*z$

### 6.29.1 Detailed Description

This module handles all Laplace equations routines.

## 6.30 LINEAR\_ELASTICITY\_ROUTINES Namespace Reference

This module handles all linear elasticity routines.

### Functions

- subroutine [LINEAR\\_ELASTICITYFINITEELEMENTCALCULATE](#) (EQUATIONS\_SET, ELEMENT\_NUMBER, ERR, ERROR,\*)
 

*Calculates the element stiffness matrices and RHS for a linear elasticity finite element equations set.*
- subroutine [LINEARELASTICITYEQUATIONSSETSETUP](#) (EQUATIONS\_SET, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)
 

*Sets up the Linear elasticity equation type of an elasticity equations set class.*
- subroutine [LINEARELASTICITYEQUATIONSSETSUBTYPESET](#) (EQUATIONS\_SET, EQUATIONS\_SET\_SUBTYPE, ERR, ERROR,\*)
 

*Sets/changes the equation subtype for a linear elasticity equation type of an elasticity equations set class.*
- subroutine [LINEARELASTICITYPROBLEMSETUP](#) (PROBLEM, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)
 

*Sets up the linear elasticity problem.*
- subroutine [LINEARELASTICITYPROBLEMSUBTYPESET](#) (PROBLEM, PROBLEM\_SUBTYPE, ERR, ERROR,\*)
 

*Sets/changes the problem subtype for a linear elasticity type .*

### 6.30.1 Detailed Description

This module handles all linear elasticity routines.

### 6.30.2 Function Documentation

#### 6.30.2.1 subroutine LINEAR\_ELASTICITY\_ROUTINES::LINEAR\_ELASTICITY\_EQUATIONS\_SET\_SETUP (TYPE(EQUATIONS\_SET\_TYPE),pointer EQUATIONS\_SET, INTEGER(INTG),intent(in) SETUP\_TYPE, INTEGER(INTG),intent(in) ACTION\_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)

Sets up the Linear elasticity equation type of an elasticity equations set class.

#### Parameters:

- EQUATIONS\_SET** A pointer to the equations set to setup a Laplace equation on.  
**SETUP\_TYPE** The setup type  
**ACTION\_TYPE** The action type  
**ERR** The error code  
**ERROR** The error string

Definition at line 122 of file linear\_elasticity\_routines.f90.

```
References BASE_ROUTINES::ENTERS(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_
MAPPING_CREATE_FINISH(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_
MAPPING_CREATE_START(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_
LINEAR_MATRICES_NUMBER_SET(), EQUATIONS_MAPPING_ROUTINES::EQUATIONS_
MAPPING_LINEAR_MATRICES_VARIABLE_TYPES_SET(), EQUATIONS_MAPPING_
ROUTINES::EQUATIONS_MAPPING_RHS_VARIABLE_TYPE_SET(), EQUATIONS_
SET_CONSTANTS::EQUATIONS_SET_BEM SOLUTION_METHOD, EQUATIONS_SET_
SET_CONSTANTS::EQUATIONS_SET_FD SOLUTION_METHOD, EQUATIONS_SET_
CONSTANTS::EQUATIONS_SET_FEM SOLUTION_METHOD, EQUATIONS_SET_
CONSTANTS::EQUATIONS_SET_FV SOLUTION_METHOD, EQUATIONS_SET_
CONSTANTS::EQUATIONS_SET_GFEM SOLUTION_METHOD, EQUATIONS_SET_
CONSTANTS::EQUATIONS_SET_GFV SOLUTION_METHOD, EQUATIONS_SET_
CONSTANTS::EQUATIONS_SET_LINEAR, EQUATIONS_SET_CONSTANTS::EQUATIONS_
SET_NO_SUBTYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_
ANALYTIC_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_
DEPENDENT_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_
EQUATIONS_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_
FINISH_ACTION, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FIXED_
CONDITIONS_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_
GEOMETRY_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_INITIAL_
TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_MATERIALS_
TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_SOURCE_TYPE,
EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_START_ACTION, EQUATIONS_
SET_CONSTANTS::EQUATIONS_SET_STATIC, BASE_ROUTINES::ERRORS(), BASE_
ROUTINES::EXITS(), FIELD_ROUTINES::FIELD_BOUNDARY_CONDITIONS_SET_TYPE,
FIELD_ROUTINES::FIELD_CREATE_FINISH(), FIELD_ROUTINES::FIELD_DEPENDENT_
TYPE, FIELD_ROUTINES::FIELD_GENERAL_TYPE, FIELD_ROUTINES::FIELD_NODE_
BASED_INTERPOLATION, FIELD_ROUTINES::FIELD_NORMAL_VARIABLE_TYPE, FIELD_
ROUTINES::FIELD_STANDARD_VARIABLE_TYPE, MATRIX_VECTOR::MATRIX_BLOCK_
STORAGE_TYPE, and MATRIX_VECTOR::MATRIX_COMPRESSED_ROW_STORAGE_TYPE.
```

Referenced by ELASTICITY\_ROUTINES::ELASTICITY\_EQUATIONS\_SET\_SETUP(), and LINEAR\_ELASTICITY\_EQUATIONS\_SET\_SUBTYPE\_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.30.2.2 subroutine LINEAR\_ELASTICITY\_ROUTINES::LINEAR\_ELASTICITY\_-
EQUATIONS\_SET\_SUBTYPE\_SET (TYPE(EQUATIONS\_SET\_TYPE),pointer
EQUATIONS\_SET, INTEGER(INTG),intent(in) EQUATIONS\_SET\_SUBTYPE,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)
[private]**

Sets/changes the equation subtype for a linear elasticity equation type of an elasticity equations set class.

#### Parameters:

**EQUATIONS\_SET** A pointer to the equations set to set the equation subtype for

**EQUATIONS\_SET\_SUBTYPE** The equation subtype to set

**ERR** The error code

**ERROR** The error string

Definition at line 379 of file linear\_elasticity\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LINEAR_ELASTICITY_CLASS`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LINEAR_ELASTICITY_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NO_SUBTYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_INITIAL_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_START_ACTION`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `LINEAR_ELASTICITY_EQUATIONS_SET_SETUP()`.

Referenced by `ELASTICITY_ROUTINES::EL()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.30.2.3 subroutine LINEAR\_ELASTICITY\_ROUTINES::LINEAR\_ELASTICITYFINITE\_ELEMENT\_CALCULATE (TYPE(EQUATIONS\_SET\_TYPE),pointer EQUATIONS\_SET, INTEGER(INTG),intent(in) ELEMENT\_NUMBER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Calculates the element stiffness matrices and RHS for a linear elasticity finite element equations set.

**Parameters:**

**EQUATIONS\_SET** A pointer to the equations set to perform the finite element calculations on  
**ELEMENT\_NUMBER** The element number to calculate  
**ERR** The error code  
**ERROR** The error string

Definition at line 88 of file linear\_elasticity\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `ELASTICITY_ROUTINES::ELASTICITYFINITE_ELEMENT_CALCULATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.30.2.4 subroutine LINEAR\_ELASTICITY\_ROUTINES::LINEAR\_ELASTICITYPROBLEM\_SETUP (TYPE(PROBLEM\_TYPE),pointer PROBLEM, INTEGER(INTG),intent(in) SETUP\_TYPE, INTEGER(INTG),intent(in) ACTION\_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets up the linear elasticity problem.

**Parameters:**

**PROBLEM** A pointer to the solutions set to setup a Laplace equation on.  
**SETUP\_TYPE** The setup type  
**ACTION\_TYPE** The action type  
**ERR** The error code  
**ERROR** The error string

Definition at line 420 of file linear\_elasticity\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_ELASTICITY_CLASS`,    `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LINEAR_ELASTICITY_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NO_SUBTYPE`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_ROUTINES::FIELD_STANDARD_VARIABLE_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_SETUP_CONTROL_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_SETUP_DO_ACTION`,    `PROBLEM_CONSTANTS::PROBLEM_SETUP_FINISH_ACTION`,    `PROBLEM_CONSTANTS::PROBLEM_SETUP_INITIAL_TYPE`,    `PROBLEM_CONSTANTS::PROBLEM_SETUP_SOLUTION_TYPE`,    `PROBLEM_CONSTANTS::PROBLEM_SETUP_SOLVER_TYPE`,    `PROBLEM_CONSTANTS::PROBLEM_SETUP_START_ACTION`,    `KINDS::PTR`,    `SOLVER_ROUTINES::SOLVER_CREATE_FINISH()`, `SOLVER_ROUTINES::SOLVER_CREATE_START()`,    `SOLVER_ROUTINES::SOLVER_LIBRARY_SET()`,    `SOLVER_ROUTINES::SOLVER_LINEAR_TYPE`,    `SOLVER_ROUTINES::SOLVER_PETSC_LIBRARY`,    `SOLVER_ROUTINES::SOLVER_SPARSE_MATRICES`,    and    `SOLVER_ROUTINES::SOLVER_SPARSITY_TYPE_SET()`.

Referenced by `ELASTICITY_ROUTINES::ELASTICITY_PROBLEM_SETUP()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.30.2.5 subroutine `LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SUBTYPE_SET`(`TYPE(PROBLEM_TYPE)`,pointer *PROBLEM*, `INTEGER(INTG),intent(in) PROBLEM_SUBTYPE`, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR`, \*)

Sets/changes the problem subtype for a linear elasticity type .

##### Parameters:

***PROBLEM*** A pointer to the problem to set the problem subtype for

***PROBLEM\_SUBTYPE*** The problem subtype to set

***ERR*** The error code

***ERROR*** The error string

Definition at line 566 of file linear\_elasticity\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `PROBLEM_CONSTANTS::PROBLEM_NO_SUBTYPE`.

Referenced by `ELASTICITY_ROUTINES::ELASTICITY_PROBLEM_CLASS_TYPE_SET()`.

Here is the call graph for this function:

Here is the caller graph for this function:

## 6.31 LISTS Namespace Reference

Implements lists of base types.

### Classes

- struct [LIST\\_PTR\\_TYPE](#)  
*Buffer type to allow arrays of pointers to a list.*
- struct [LIST\\_TYPE](#)  
*Contains information on a list.*
- interface [LIST\\_ITEM\\_ADD](#)  
*Adds an item to the end of a list.*
- interface [LIST\\_ITEM\\_IN\\_LIST](#)  
*Determines if an item is in a list and returns the position of the item.*
- interface [LIST\\_DETACH\\_AND\\_DESTROY](#)  
*Detaches the list values from a list and returns them as a pointer to a array of base type before destroying the list.*
- interface [LIST\\_SEARCH](#)  
*Searches a list for a given value and returns the position in the list if the value exists.*
- interface [LIST\\_SEARCH\\_LINEAR](#)  
*Searches a list using the linear search method.*
- interface [LIST\\_SORT](#)  
*Sorts a list into ascending order.*
- interface [LIST\\_SORT\\_BUBBLE](#)  
*Sorts a list into assending order using the bubble sort method.*
- interface [LIST\\_SORT\\_HEAP](#)  
*Sorts a list into assending order using the heap sort method.*
- interface [LIST\\_SORT\\_SHELL](#)  
*Sorts a list into either assending or descending order using the shell sort method.*

### Functions

- subroutine [LIST\\_CREATE\\_FINISH](#) (LIST, ERR, ERROR,\*)  
*Finishes the creation of a list created with LIST\_CREATE\_START.*
- subroutine [LIST\\_CREATE\\_START](#) (LIST, ERR, ERROR,\*)  
*Starts the creation of a list and returns a pointer to the created list.*

- subroutine [LIST\\_DATA\\_TYPE\\_SET](#) (LIST, DATA\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the data type for a list.*
- subroutine [LIST\\_DESTROY](#) (LIST, ERR, ERROR,\*)
 

*Destroys a list.*
- subroutine [LIST\\_FINALISE](#) (LIST, ERR, ERROR,\*)
 

*Finalises a list and deallocates all memory.*
- subroutine [LIST\\_INITIALISE](#) (LIST, ERR, ERROR,\*)
 

*Initialises a list and all its components.*
- subroutine [LIST\\_INITIAL\\_SIZE\\_SET](#) (LIST, INITIAL\_SIZE, ERR, ERROR,\*)
 

*Sets/changes the initial size for a list.*
- subroutine [LIST\\_ITEM\\_ADD\\_INTG1](#) (LIST, ITEM, ERR, ERROR,\*)
 

*Adds an item to the end of an integer list.*
- subroutine [LIST\\_ITEM\\_ADD\\_SP1](#) (LIST, ITEM, ERR, ERROR,\*)
 

*Adds an item to the end of a single precision real list.*
- subroutine [LIST\\_ITEM\\_ADD\\_DP1](#) (LIST, ITEM, ERR, ERROR,\*)
 

*Adds an item to the end of a double precision real list.*
- subroutine [LIST\\_ITEM\\_IN\\_LIST\\_INTG1](#) (LIST, ITEM, LIST\_ITEM, ERR, ERROR,\*)
 

*Determines if ITEM is in the given integer LIST. If it is LIST\_ITEM is the index in the list. If not LIST\_ITEM is 0.*
- subroutine [LIST\\_ITEM\\_IN\\_LIST\\_SP1](#) (LIST, ITEM, LIST\_ITEM, ERR, ERROR,\*)
 

*Determines if ITEM is in the given single precision real LIST. If it is LIST\_ITEM is the index in the list. If not LIST\_ITEM is 0.*
- subroutine [LIST\\_ITEM\\_IN\\_LIST\\_DP1](#) (LIST, ITEM, LIST\_ITEM, ERR, ERROR,\*)
 

*Determines if ITEM is in the given double precision real LIST. If it is LIST\_ITEM is the index in the list. If not LIST\_ITEM is 0.*
- subroutine [LIST\\_ITEM\\_DELETE](#) (LIST, LIST\_ITEM, ERR, ERROR,\*)
 

*Deletes the item given by the LIST\_ITEM index from the given list.*
- subroutine [LIST\\_NUMBER\\_OF\\_ITEMS\\_GET](#) (LIST, NUMBER\_OF\_ITEMS, ERR, ERROR,\*)
 

*Gets the current number of items in a list.*
- subroutine [LIST\\_DETACH\\_AND\\_DESTROY\\_INTG](#) (LIST, NUMBER\_IN\_LIST, LIST\_VALUES, ERR, ERROR,\*)
 

*Detaches the list values from an integer list and returns them as a pointer to a array of base type before destroying the list. The LIST\_VALUES pointer must not be associated on entry. It is up to the user to then deallocate the returned list memory.*
- subroutine [LIST\\_DETACH\\_AND\\_DESTROY\\_SP](#) (LIST, NUMBER\_IN\_LIST, LIST\_VALUES, ERR, ERROR,\*)

*Detaches the list values from a single precision real list and returns them as a pointer to a array of base type before destroying the list. The LIST\_VALUES pointer must not be associated on entry. It is up to the user to then deallocate the returned list memory.*

- subroutine [LIST\\_DETACH\\_AND\\_DESTROY\\_DP](#) (LIST, NUMBER\_IN\_LIST, LIST\_VALUES, ERR, ERROR,\*)

*Detaches the list values from a double precision real list and returns them as a pointer to a array of base type before destroying the list. The LIST\_VALUES pointer must not be associated on entry. It is up to the user to then deallocate the returned list memory.*

- subroutine [LIST\\_REMOVE\\_DUPLICATES](#) (LIST, ERR, ERROR,\*)

*Removes duplicate entries from a list. A side effect of this is that the list is sorted.*

- subroutine [LIST\\_SEARCH\\_INTG\\_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,\*)

*Searches an integer array list A for VALUE. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.*

- subroutine [LIST\\_SEARCH\\_SP\\_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,\*)

*Searches a single precision real array list A for VALUE. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.*

- subroutine [LIST\\_SEARCH\\_DP\\_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,\*)

*Searches a double precision real array list A for VALUE. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.*

- subroutine [LIST\\_SEARCH\\_LINEAR\\_INTG\\_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,\*)

*Searches an integer array list A for VALUE using the linear search method. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.*

- subroutine [LIST\\_SEARCH\\_LINEAR\\_SP\\_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,\*)

*Searches a single precision real array list A for VALUE using the linear search method. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.*

- subroutine [LIST\\_SEARCH\\_LINEAR\\_DP\\_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,\*)

*Searches a double precision real array list A for VALUE using the linear search method. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.*

- subroutine [LIST\\_SORT\\_INTG\\_ARRAY](#) (A, ERR, ERROR,\*)

*Sorts an integer array list into ascending order.*

- subroutine [LIST\\_SORT\\_SP\\_ARRAY](#) (A, ERR, ERROR,\*)

*Sorts an single precision array list into ascending order.*

- subroutine [LIST\\_SORT\\_DP\\_ARRAY](#) (A, ERR, ERROR,\*)

*Sorts an double precision array list into ascending order.*

- subroutine [LIST\\_SORT\\_BUBBLE\\_INTG\\_ARRAY](#) (A, ERR, ERROR,\*)

*BUBBLE\_SORT\_INTG performs a bubble sort on an integer array list.*

- subroutine [LIST\\_SORT\\_BUBBLE\\_SP\\_ARRAY](#) (A, ERR, ERROR,\*)

*BUBBLE\_SORT\_SP performs a bubble sort on a single precision array list.*

- subroutine [LIST\\_SORT\\_BUBBLE\\_DP\\_ARRAY](#) (A, ERR, ERROR,\*)
 

*BUBBLE\_SORT\_DP* performs a bubble sort on a double precision list.
- subroutine [LIST\\_SORT\\_HEAP\\_INTG\\_ARRAY](#) (A, ERR, ERROR,\*)
 

*Sorts an integer array list into assending order using the heap sort method.*
- subroutine [LIST\\_SORT\\_HEAP\\_SP\\_ARRAY](#) (A, ERR, ERROR,\*)
 

*Sorts a real single precision array list into assending order using the heap sort method.*
- subroutine [LIST\\_SORT\\_HEAP\\_DP\\_ARRAY](#) (A, ERR, ERROR,\*)
 

*Sorts a real double precision array list into assending order using the heap sort method.*
- subroutine [LIST\\_SORT\\_SHELL\\_INTG\\_ARRAY](#) (A, ERR, ERROR,\*)
 

*Sorts an integer array list into either assending or descending order using the shell sort method.*
- subroutine [LIST\\_SORT\\_SHELL\\_SP\\_ARRAY](#) (A, ERR, ERROR,\*)
 

*Sorts a real single precision array list into either assending or descending order using the shell sort method.*
- subroutine [LIST\\_SORT\\_SHELL\\_DP\\_ARRAY](#) (A, ERR, ERROR,\*)
 

*Sorts a real double precision array list into either assending or descending order using the shell sort method.*

## Variables

- INTEGER(INTG), parameter [LIST\\_INTG\\_TYPE](#) = INTEGER\_TYPE
 

*Integer data type for a list.*
- INTEGER(INTG), parameter [LIST\\_SP\\_TYPE](#) = SINGLE\_REAL\_TYPE
 

*Single precision real data type for a list.*
- INTEGER(INTG), parameter [LIST\\_DP\\_TYPE](#) = DOUBLE\_REAL\_TYPE
 

*Double precision real data type for a list.*
- INTEGER(INTG), parameter [LIST\\_UNSORTED\\_TYPE](#) = 1
 

*Unsorted list type.*
- INTEGER(INTG), parameter [LIST\\_SORT\\_ASCENDING\\_TYPE](#) = 2
 

*Ascending order for sort.*
- INTEGER(INTG), parameter [LIST\\_SORT\\_DESCENDING\\_TYPE](#) = 3
 

*Descending order for sort.*
- INTEGER(INTG), parameter [LIST\\_BUBBLE\\_SORT\\_METHOD](#) = 1
 

*Bubble sort method.*
- INTEGER(INTG), parameter [LIST\\_SHELL\\_SORT\\_METHOD](#) = 2
 

*Shell sort method.*
- INTEGER(INTG), parameter [LIST\\_HEAP\\_SORT\\_METHOD](#) = 3
 

*Heap sort method.*

### 6.31.1 Detailed Description

Implements lists of base types.

### 6.31.2 Function Documentation

#### 6.31.2.1 subroutine LISTS::LIST\_CREATE\_FINISH (TYPE(List\_Type),pointer *LIST*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Finishes the creation of a list created with LIST\_CREATE\_START.

See also:

[LISTS::LIST\\_CREATE\\_START](#).

Parameters:

*LIST* A pointer to the list to finish

*ERR* The error code

*ERROR* The error string.

Definition at line 192 of file lists.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_-ROUTINES::EXITS(), LIST\_DP\_TYPE, LIST\_INTG\_TYPE, and LIST\_SP\_TYPE.

Referenced by    MESH\_ROUTINES::DECOMPOSITION\_NUMBER\_OF\_DOMAINS\_SET(), DOMAIN\_MAPPINGS::DOMAIN\_MAPPINGS\_LOCAL\_FROM\_GLOBAL\_CALCULATE(), FIELD\_ROUTINES::FIELD\_INTERPOLATION\_PARAMETERS\_LINE\_GET(),      MESH\_-ROUTINES::MESH\_TOPOLOGY\_ELEMENTS\_ADJACENT\_ELEMENTS\_CALCULATE(), MESH\_-ROUTINES::MESH\_TOPOLOGY\_NODES\_DERIVATIVES\_CALCULATE(),      and      SOLVER\_-MATRICES\_ROUTINES::SOLVER\_MATRIX\_STRUCTURE\_CALCULATE().

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.31.2.2 subroutine LISTS::LIST\_CREATE\_START (TYPE(List\_Type),pointer *LIST*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Starts the creation of a list and returns a pointer to the created list.

See also:

[LISTS::LIST\\_CREATE\\_FINISH](#).

Parameters:

*LIST* On exit, pointer to the list to create. Must not be associated on entry.

*ERR* The error code.

*ERROR* The error string.

Definition at line 243 of file lists.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `LIST_FINALISE()`,    `LIST_HEAP_SORT_METHOD`,    `LIST_INITIALISE()`,    `LIST_INTG_TYPE`, and `LIST_SORT_ASCENDING_TYPE`.

Referenced    by    `MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET()`,  
`DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE()`,  
`FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`,                         `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE()`, `MESH_ROUTINES::MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE()`,    and    `SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_CALCULATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### **6.31.2.3 subroutine `LISTS::LIST_DATA_TYPE_SET` (`TYPE(List_Type)`,pointer *LIST*, `INTEGER(INTG),intent(in) DATA_TYPE`, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, *`)**

Sets/changes the data type for a list.

##### **Parameters:**

*LIST* A pointer to the list

*DATA\_TYPE* The data type of the list to set

**See also:**

`LISTS::DataType,LISTS`

*ERR* The error code

*ERROR* The error string

Definition at line 279 of file lists.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `LIST_DP_TYPE`, `LIST_INTG_TYPE`, and `LIST_SP_TYPE`.

Referenced    by    `MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET()`,  
`DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE()`,  
`FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`,                         `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE()`, `MESH_ROUTINES::MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE()`,    and    `SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_CALCULATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### **6.31.2.4 subroutine `LISTS::LIST_DESTROY` (`TYPE(List_Type)`,pointer *LIST*, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, *`)**

Destroys a list.

##### **Parameters:**

*LIST* A pointer to the list to destroy

*ERR* The error code

**ERROR** The error string

Definition at line 323 of file lists.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `LIST_FINALISE()`.

Referenced by `MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET()`, `DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE()`, `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`, `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE()`, `MESH_ROUTINES::MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE()`, and `SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_CALCULATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.31.2.5 subroutine `LISTS::LIST_DETACH_AND_DESTROY_DP` (`TYPE(LIST_TYPE)`,pointer `LIST`, `INTEGER(INTG),intent(out) NUMBER_IN_LIST`,  
`REAL(DP),dimension(:),pointer LIST_VALUES, INTEGER(INTG),intent(out) ERR`,  
`TYPE(VARYING_STRING),intent(out) ERROR, */ [private]`**

Detaches the list values from a double precision real list and returns them as a pointer to a array of base type before destroying the list. The `LIST_VALUES` pointer must not be associated on entry. It is up to the user to then deallocate the returned list memory.

#### Parameters:

**`LIST`** The pointer to the list

**`NUMBER_IN_LIST`** On exit, the number in the list that has been detached.

**`LIST_VALUES`** On exit, a pointer to the detached list. Must not be associated on entry.

**`ERR`** The error code

**`ERROR`** The error string

Definition at line 931 of file lists.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `LIST_DP_TYPE`, and `LIST_FINALISE()`.

Here is the call graph for this function:

**6.31.2.6 subroutine `LISTS::LIST_DETACH_AND_DESTROY_INTG` (`TYPE(LIST_TYPE)`,pointer `LIST`, `INTEGER(INTG),intent(out) NUMBER_IN_LIST`,  
`INTEGER(INTG),dimension(:),pointer LIST_VALUES, INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, */ [private]`**

Detaches the list values from an integer list and returns them as a pointer to a array of base type before destroying the list. The `LIST_VALUES` pointer must not be associated on entry. It is up to the user to then deallocate the returned list memory.

#### Parameters:

**`LIST`** The pointer to the list

**NUMBER\_IN\_LIST** On exit, the number in the list that has been detached.

**LIST\_VALUES** On exit, a pointer to the detached list. Must not be associated on entry.

**ERR** The error code

**ERROR** The error string

Definition at line 830 of file lists.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `LIST_FINALISE()`, and `LIST_INTG_TYPE`.

Here is the call graph for this function:

**6.31.2.7 subroutine LISTS::LIST\_DETACH\_AND\_DESTROY\_SP (TYPE(List\_Type),pointer LIST, INTEGER(INTG),intent(out) NUMBER\_IN\_LIST, REAL(SP),dimension(:),pointer LIST\_VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Detaches the list values from a single precision real list and returns them as a pointer to a array of base type before destroying the list. The `LIST_VALUES` pointer must not be associated on entry. It is up to the user to then deallocate the returned list memory.

#### Parameters:

**LIST** The pointer to the list

**NUMBER\_IN\_LIST** On exit, the number in the list that has been detached.

**LIST\_VALUES** On exit, a pointer to the detached list. Must not be associated on entry.

**ERR** The error code

**ERROR** The error string

Definition at line 881 of file lists.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `LIST_FINALISE()`, and `LIST_SP_TYPE`.

Here is the call graph for this function:

**6.31.2.8 subroutine LISTS::LIST\_FINALISE (TYPE(List\_Type),pointer LIST, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Finalises a list and deallocates all memory.

#### Parameters:

**LIST** A pointer to the list to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 351 of file lists.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by LIST\_CREATE\_START(), LIST\_DESTROY(), LIST\_DETACH\_AND\_DESTROY\_DP(), LIST\_DETACH\_AND\_DESTROY\_INTG(), and LIST\_DETACH\_AND\_DESTROY\_SP().

Here is the call graph for this function:

Here is the caller graph for this function:

#### **6.31.2.9 subroutine LISTS::LIST\_INITIAL\_SIZE\_SET (TYPE(LIST\_TYPE),pointer *LIST*, INTEGER(INTG),intent(in) *INITIAL\_SIZE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Sets/changes the initial size for a list.

##### **Parameters:**

*LIST* A pointer to the list

*INITIAL\_SIZE* The initial size of the list to set. Must be greater than zero.

*ERR* The error code

*ERROR* The error string

Definition at line 417 of file lists.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by MESH\_ROUTINES::DECOMPOSITION\_NUMBER\_OF\_DOMAINS\_SET(), DOMAIN\_MAPPINGS::DOMAIN\_MAPPINGS\_LOCAL\_FROM\_GLOBAL\_CALCULATE(), FIELD\_ROUTINES::FIELD\_INTERPOLATION\_PARAMETERS\_LINE\_GET(), MESH\_ROUTINES::MESH\_TOPOLOGY\_ELEMENTS\_ADJACENT\_ELEMENTS\_CALCULATE(), MESH\_ROUTINES::MESH\_TOPOLOGY\_NODES\_DERIVATIVES\_CALCULATE(), and SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRIX\_STRUCTURE\_CALCULATE().

Here is the call graph for this function:

Here is the caller graph for this function:

#### **6.31.2.10 subroutine LISTS::LIST\_INITIALISE (TYPE(LIST\_TYPE),pointer *LIST*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Initialises a list and all its components.

##### **Parameters:**

*LIST* A pointer to the list to initialise

*ERR* The error code

*ERROR* The error string

Definition at line 380 of file lists.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by LIST\_CREATE\_START().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.31.2.11 subroutine LISTS::LIST\_ITEM\_ADD\_DP1 (TYPE(LIST\_TYPE),pointer  
 $LIST$ , REAL(DP),intent(in)  $ITEM$ , INTEGER(INTG),intent(out)  $ERR$ ,  
 TYPE(VARYING\_STRING),intent(out)  $ERROR$ , \*) [private]**

Adds an item to the end of a double precision real list.

**Parameters:**

**$LIST$**  A pointer to the list

**$ITEM$**  The item to add

**$ERR$**  The error code

**$ERROR$**  The error string

Definition at line 566 of file lists.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_-ROUTINES::EXITS(), and LIST\_DP\_TYPE.

Here is the call graph for this function:

**6.31.2.12 subroutine LISTS::LIST\_ITEM\_ADD\_INTG1 (TYPE(LIST\_TYPE),pointer  
 $LIST$ , INTEGER(INTG),intent(in)  $ITEM$ , INTEGER(INTG),intent(out)  $ERR$ ,  
 TYPE(VARYING\_STRING),intent(out)  $ERROR$ , \*) [private]**

Adds an item to the end of an integer list.

**Parameters:**

**$LIST$**  A pointer to the list

**$ITEM$**  The item to add

**$ERR$**  The error code

**$ERROR$**  The error string

Definition at line 457 of file lists.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_-ROUTINES::EXITS(), and LIST\_INTG\_TYPE.

Here is the call graph for this function:

**6.31.2.13 subroutine LISTS::LIST\_ITEM\_ADD\_SP1 (TYPE(LIST\_TYPE),pointer  
 $LIST$ , REAL(SP),intent(in)  $ITEM$ , INTEGER(INTG),intent(out)  $ERR$ ,  
 TYPE(VARYING\_STRING),intent(out)  $ERROR$ , \*) [private]**

Adds an item to the end of a single precision real list.

**Parameters:**

**$LIST$**  A pointer to the list

**$ITEM$**  The item to add

**$ERR$**  The error code

**$ERROR$**  The error string

Definition at line 511 of file lists.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_-ROUTINES::EXITS(), and LIST\_SP\_TYPE.

Here is the call graph for this function:

**6.31.2.14 subroutine LISTS::LIST\_ITEM\_DELETE (TYPE(LIST\_TYPE),pointer *LIST*,  
INTEGER(INTG),intent(in) *LIST\_ITEM*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Deletes the item given by the *LIST\_ITEM* index from the given list.

**Parameters:**

*LIST* The pointer to the list

*LIST\_ITEM* The position in the list to delete.

*ERR* The error code

*ERROR* The error string

Definition at line 749 of file lists.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_-ROUTINES::EXITS(), LIST\_DP\_TYPE, LIST\_INTG\_TYPE, and LIST\_SP\_TYPE.

Here is the call graph for this function:

**6.31.2.15 subroutine LISTS::LIST\_ITEM\_IN\_LIST\_DP1 (TYPE(LIST\_TYPE),pointer  
*LIST*, REAL(DP),intent(in) *ITEM*, INTEGER(INTG),intent(out) *LIST\_ITEM*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
\*) [private]**

Determines if *ITEM* is in the given double precision real *LIST*. If it is *LIST\_ITEM* is the index in the list. If not *LIST\_ITEM* is 0.

**Parameters:**

*LIST* The pointer to the list

*ITEM* The item to find.

*LIST\_ITEM* On exit, the position of the item in the list. If the item does not exist then the value of 0 is returned.

*ERR* The error code

*ERROR* The error string

Definition at line 707 of file lists.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_-ROUTINES::EXITS(), and LIST\_DP\_TYPE.

Here is the call graph for this function:

---

**6.31.2.16 subroutine LISTS::LIST\_ITEM\_IN\_LIST\_INTG1** (TYPE(LIST\_TYPE),pointer  
*LIST*, INTEGER(INTG),intent(in) *ITEM*, INTEGER(INTG),intent(out) *LIST\_ITEM*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Determines if *ITEM* is in the given integer *LIST*. If it is *LIST\_ITEM* is the index in the list. If not *LIST\_ITEM* is 0.

**Parameters:**

*LIST* The pointer to the list

*ITEM* The item to find.

*LIST\_ITEM* On exit, the position of the item in the list. If the item does not exist then the value of 0 is returned.

*ERR* The error code

*ERROR* The error string.

Definition at line 621 of file lists.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_-ROUTINES::EXITS(), and LIST\_INTG\_TYPE.

Here is the call graph for this function:

**6.31.2.17 subroutine LISTS::LIST\_ITEM\_IN\_LIST\_SP1** (TYPE(LIST\_TYPE),pointer  
*LIST*, REAL(SP),intent(in) *ITEM*, INTEGER(INTG),intent(out) *LIST\_ITEM*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Determines if *ITEM* is in the given single precision real *LIST*. If it is *LIST\_ITEM* is the index in the list. If not *LIST\_ITEM* is 0.

**Parameters:**

*LIST* The pointer to the list

*ITEM* The item to find.

*LIST\_ITEM* On exit, the position of the item in the list. If the item does not exist then the value of 0 is returned.

*ERR* The error code

*ERROR* The error string

Definition at line 664 of file lists.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_-ROUTINES::EXITS(), and LIST\_SP\_TYPE.

Here is the call graph for this function:

**6.31.2.18 subroutine LISTS::LIST\_NUMBER\_OF\_ITEMS\_GET** (TYPE(LIST\_TYPE),pointer  
*LIST*, INTEGER(INTG),intent(out) *NUMBER\_OF\_ITEMS*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*)

Gets the current number of items in a list.

**Parameters:**

**LIST** A pointer to the list

**NUMBER\_OF\_ITEMS** On exit, the current number of items in the list

**ERR** The error code

**ERROR** The error string

Definition at line 796 of file lists.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    and    **BASE\_ROUTINES::EXITS()**.

Referenced    by    **SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRIX\_STRUCTURE\_CALCULATE()**.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.31.2.19 subroutine LISTS::LIST\_REMOVE\_DUPLICATES (TYPE(LIST\_TYPE),pointer LIST, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Removes duplicate entries from a list. A side effect of this is that the list is sorted.

**Parameters:**

**LIST** The pointer to the list

**ERR** The error code

**ERROR** The error string

Definition at line 981 of file lists.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    **BASE\_ROUTINES::EXITS()**, **LIST\_DP\_TYPE**, **LIST\_INTG\_TYPE**, and **LIST\_SP\_TYPE**.

Referenced    by    **MESH\_ROUTINES::DECOMPOSITION\_NUMBER\_OF\_DOMAINS\_SET()**, **DOMAIN\_MAPPINGS::DOMAIN\_MAPPINGS\_LOCAL\_FROM\_GLOBAL\_CALCULATE()**, **FIELD\_ROUTINES::FIELD\_INTERPOLATION\_PARAMETERS\_LINE\_GET()**,    **MESH\_ROUTINES::MESH\_TOPOLOGY\_ELEMENTS\_ADJACENT\_ELEMENTS\_CALCULATE()**, **MESH\_ROUTINES::MESH\_TOPOLOGY\_NODES\_DERIVATIVES\_CALCULATE()**,    and    **SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRIX\_STRUCTURE\_CALCULATE()**.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.31.2.20 subroutine LISTS::LIST\_SEARCH\_DP\_ARRAY (REAL(DP),dimension(:),intent(in) A, REAL(DP),intent(in) VALUE, INTEGER(INTG),intent(out) POSITION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Searches a double precision real array list A for VALUE. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

**Parameters:****A** The list to search**VALUE** The value to search for**POSITION** On exit, the position of value in the list. If value does not exist in the list the returned position is zero.**ERR** The error code**ERROR** The error string

Definition at line 1157 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.31.2.21 subroutine LISTS::LIST_SEARCH_INTG_ARRAY (INTEGER(INTG),dimension(:),intent(in) A, INTEGER(INTG),intent(in) VALUE, INTEGER(INTG),intent(out) POSITION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Searches an integer array list A for VALUE. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

**Parameters:****A** The list to search**VALUE** The value to search for**POSITION** On exit, the position of value in the list. If value does not exist in the list the returned position is zero.**ERR** The error code**ERROR** The error string

Definition at line 1103 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.31.2.22 subroutine LISTS::LIST_SEARCH_LINEAR_DP_ARRAY (REAL(DP),dimension(:),intent(in) A, REAL(DP),intent(in) VALUE, INTEGER(INTG),intent(out) POSITION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Searches a double precision real array list A for VALUE using the linear search method. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

**Parameters:****A** The list to search**VALUE** The value to search for

**POSITION** On exit, the position of value in the list. If value does not exist in the list the returned position is zero.

**ERR** The error code

**ERROR** The error string

Definition at line 1266 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.31.2.23 subroutine LISTS::LIST\_SEARCH\_LINEAR\_INTG\_ARRAY**  
`(INTEGER(INTG),dimension(:),intent(in) A, INTEGER(INTG),intent(in) VALUE,`  
`INTEGER(INTG),intent(out) POSITION, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Searches an integer array list A for VALUE using the linear search method. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

#### Parameters:

**A** The list to search

**VALUE** The value to search for

**POSITION** On exit, the position of value in the list. If value does not exist in the list the returned position is zero.

**ERR** The error code

**ERROR** The error string

Definition at line 1184 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.31.2.24 subroutine LISTS::LIST\_SEARCH\_LINEAR\_SP\_ARRAY**  
`(REAL(SP),dimension(:),intent(in) A, REAL(SP),intent(in) VALUE,`  
`INTEGER(INTG),intent(out) POSITION, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Searches a single precision real array list A for VALUE using the linear search method. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

#### Parameters:

**A** The list to search

**VALUE** The value to search for

**POSITION** On exit, the position of value in the list. If value does not exist in the list the returned position is zero.

**ERR** The error code

**ERROR** The error string

Definition at line 1225 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.31.2.25 subroutine LISTS::LIST_SEARCH_SP_ARRAY (REAL(SP),dimension(:),intent(in) A, REAL(SP),intent(in) VALUE, INTEGER(INTG),intent(out) POSITION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Searches a single precision real array list `A` for `VALUE`. If the search is successful `POSITION` contains the index of the position of `VALUE` in the list otherwise `POSITION` is zero.

#### Parameters:

`A` The list to search

`VALUE` The value to search for

`POSITION` On exit, the position of value in the list. If value does not exist in the list the returned position is zero.

`ERR` The error code

`ERROR` The error string

Definition at line 1130 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.31.2.26 subroutine LISTS::LIST_SORT_BUBBLE_DP_ARRAY (REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

`BUBBLE_SORT_DP` performs a bubble sort on a double precision list.

#### Parameters:

`A` The list to sort

`ERR` The error code

`ERROR` The error string

Definition at line 1463 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.31.2.27 subroutine LISTS::LIST_SORT_BUBBLE_INTG_ARRAY (INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

`BUBBLE_SORT_INTG` performs a bubble sort on an integer array list.

**Parameters:**

- A** The list to sort
- ERR** The error code
- ERROR** The error string

Definition at line 1382 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.31.2.28 subroutine LISTS::LIST\_SORT\_BUBBLE\_SP\_ARRAY**  
`(REAL(SP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

`BUBBLE_SORT_SP` performs a bubble sort on a single precision array list.

**Parameters:**

- A** The list to sort
- ERR** The error code
- ERROR** The error string

Definition at line 1422 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.31.2.29 subroutine LISTS::LIST\_SORT\_DP\_ARRAY** `(REAL(DP),dimension(:),intent(inout) A,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`  
`*) [private]`

Sorts an double precision array list into ascending order.

**Parameters:**

- A** The list to sort
- ERR** The error code
- ERROR** The error string

Definition at line 1357 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

---

**6.31.2.30 subroutine LISTS::LIST\_SORT\_HEAP\_DP\_ARRAY**  
`(REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Sorts a real double precision array list into assending order using the heap sort method.

**Parameters:**

- A** The list to sort
- ERR** The error code
- ERROR** The error string

Definition at line 1619 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.31.2.31 subroutine LISTS::LIST\_SORT\_HEAP\_INTG\_ARRAY**  
`(INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Sorts an integer array list into assending order using the heap sort method.

**Parameters:**

- A** The list to sort
- ERR** The error code
- ERROR** The error string

Definition at line 1504 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.31.2.32 subroutine LISTS::LIST\_SORT\_HEAP\_SP\_ARRAY**  
`(REAL(SP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Sorts a real single precision array list into assending order using the heap sort method.

**Parameters:**

- A** The list to sort
- ERR** The error code
- ERROR** The error string

Definition at line 1561 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

---

**6.31.2.33 subroutine LISTS::LIST\_SORT\_INTG\_ARRAY (INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Sorts an integer array list into ascending order.

**Parameters:**

- A** The list to sort
- ERR** The error code
- ERROR** The error string

Definition at line 1307 of file lists.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.31.2.34 subroutine LISTS::LIST\_SORT\_SHELL\_DP\_ARRAY (REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Sorts a real double precision array list into either assending or descending order using the shell sort method.

**Parameters:**

- ERR** The error code
- ERROR** The error string

Definition at line 1760 of file lists.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.31.2.35 subroutine LISTS::LIST\_SORT\_SHELL\_INTG\_ARRAY (INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Sorts an integer array list into either assending or descending order using the shell sort method.

**Parameters:**

- A** The list to sort
- ERR** The error code
- ERROR** The error string

Definition at line 1677 of file lists.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

---

**6.31.2.36 subroutine LISTS::LIST\_SORT\_SHELL\_SP\_ARRAY**  
**(REAL(SP),dimension(:),intent(inout) *A*, INTEGER(INTG),intent(out) *ERR*,**  
**TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Sorts a real single precision array list into either assending or descending order using the shell sort method.

**Parameters:**

- A*** The list to sort
- ERR*** The error code
- ERROR*** The error string

Definition at line 1718 of file lists.f90.

References   BASE\_ROUTINES::ENTERS(),   BASE\_ROUTINES::ERRORS(),   and   BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.31.2.37 subroutine LISTS::LIST\_SORT\_SP\_ARRAY (REAL(SP),dimension(:),intent(inout) *A*,**  
**INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,**  
**\*) [private]**

Sorts an single precision array list into ascending order.

**Parameters:**

- A*** The list to sort
- ERR*** The error code
- ERROR*** The error string

Definition at line 1332 of file lists.f90.

References   BASE\_ROUTINES::ENTERS(),   BASE\_ROUTINES::ERRORS(),   and   BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

## 6.32 MACHINE\_CONSTANTS Namespace Reference

This module contains all machine dependent constants for AIX systems.

### Variables

- INTEGER(INTG), parameter `MACHINE_TYPE` = IBM\_COMPUTER
- INTEGER(INTG), parameter `MACHINE_OS` = AIX\_OS
- INTEGER(INTG), parameter `MACHINE_ENDIAN` = LITTLE\_ENDIAN\_NUMBER
- INTEGER(INTG), parameter `MACHINE_CHAR_FORMAT` = ASCII\_CHARACTER
- INTEGER(INTG), parameter `MACHINE_INT_FORMAT` = TWOS\_COMPLEMENT\_INTEGER
- INTEGER(INTG), parameter `MACHINE_SP_FORMAT` = SPIEEE\_NUMBER
- INTEGER(INTG), parameter `MACHINE_DP_FORMAT` = DPIEEE\_NUMBER
- INTEGER(INTG), parameter `INTEGER_SIZE` = 4
- INTEGER(INTG), parameter `SHORT_INTEGER_SIZE` = 2
- INTEGER(INTG), parameter `LONG_INTEGER_SIZE` = 8
- INTEGER(INTG), parameter `SINGLE_REAL_SIZE` = 4
- INTEGER(INTG), parameter `DOUBLE_REAL_SIZE` = 8
- INTEGER(INTG), parameter `CHARACTER_SIZE` = 1
- INTEGER(INTG), parameter `LOGICAL_SIZE` = 4
- INTEGER(INTG), parameter `SINGLE_COMPLEX_SIZE` = 8
- INTEGER(INTG), parameter `DOUBLE_COMPLEX_SIZE` = 16
- CHARACTER(LEN=1), parameter `ERROR_SEPARATOR_CONSTANT` = CHAR(6)

### 6.32.1 Detailed Description

This module contains all machine dependent constants for AIX systems.

This module contains all machine dependent constants for Win32 systems.

This module contains all machine dependent constants for VMS systems.

This module contains all machine dependent constants for Linux systems.

This module contains all machine dependent constants for IRIX systems.

### 6.32.2 Variable Documentation

#### 6.32.2.1 INTEGER(INTG),parameter MACHINE\_CONSTANTS::CHARACTER\_SIZE = 1

Definition at line 66 of file machine\_constants\_aix.f90.

Referenced by `BINARY_FILE::OPEN_CMISS_BINARY_FILE()`, and `BINARY_FILE::SKIP_CM_BINARY_HEADER()`.

#### 6.32.2.2 INTEGER(INTG),parameter MACHINE\_CONSTANTS::DOUBLE\_COMPLEX\_SIZE = 16

Definition at line 69 of file machine\_constants\_aix.f90.

Referenced by `BINARY_FILE::OPEN_CMISS_BINARY_FILE()`.

**6.32.2.3 INTEGER(INTG),parameter MACHINE\_CONSTANTS::DOUBLE\_REAL\_SIZE = 8**

Definition at line 65 of file machine\_constants\_aix.f90.

Referenced by BINARY\_FILE::OPEN\_CMISS\_BINARY\_FILE().

**6.32.2.4 CHARACTER(LEN=1),parameter MACHINE\_CONSTANTS::ERROR\_SEPARATOR\_CONSTANT = CHAR(6)**

Definition at line 71 of file machine\_constants\_aix.f90.

Referenced by CMISS::CMISS\_WRITE\_ERROR(), and BASE\_ROUTINES::ERRORS().

**6.32.2.5 INTEGER(INTG),parameter MACHINE\_CONSTANTS::INTEGER\_SIZE = 4**

Definition at line 61 of file machine\_constants\_aix.f90.

Referenced by F90C::F2CSTRING(), BINARY\_FILE::OPEN\_CMISS\_BINARY\_FILE(), BINARY\_FILE::RESET\_BINARY\_NUMBER\_TAGS(), and BINARY\_FILE::SKIP\_CM\_BINARY\_HEADER().

**6.32.2.6 INTEGER(INTG),parameter MACHINE\_CONSTANTS::LOGICAL\_SIZE = 4**

Definition at line 67 of file machine\_constants\_aix.f90.

Referenced by BINARY\_FILE::OPEN\_CMISS\_BINARY\_FILE().

**6.32.2.7 INTEGER(INTG),parameter MACHINE\_CONSTANTS::LONG\_INTEGER\_SIZE = 8**

Definition at line 63 of file machine\_constants\_aix.f90.

Referenced by BINARY\_FILE::OPEN\_CMISS\_BINARY\_FILE().

**6.32.2.8 INTEGER(INTG),parameter MACHINE\_CONSTANTS::MACHINE\_CHAR\_FORMAT = ASCII\_CHARACTER**

Definition at line 57 of file machine\_constants\_aix.f90.

Referenced by BINARY\_FILE::OPEN\_CMISS\_BINARY\_FILE().

**6.32.2.9 INTEGER(INTG),parameter MACHINE\_CONSTANTS::MACHINE\_DP\_FORMAT = DPIEEE\_NUMBER**

Definition at line 60 of file machine\_constants\_aix.f90.

Referenced by BINARY\_FILE::OPEN\_CMISS\_BINARY\_FILE().

**6.32.2.10 INTEGER(INTG),parameter MACHINE\_CONSTANTS::MACHINE\_ENDIAN = LITTLE\_ENDIAN\_NUMBER**

Definition at line 56 of file machine\_constants\_aix.f90.

Referenced by BINARY\_FILE::OPEN\_BINARY\_FILE(), BINARY\_FILE::OPEN\_CMISS\_BINARY\_FILE(), BINARY\_FILE::READ\_BINARY\_FILE\_DP(), BINARY\_FILE::READ\_BINARY\_FILE\_DP1(), BINARY\_FILE::READ\_BINARY\_FILE\_DPC(), BINARY\_FILE::READ\_BINARY\_FILE\_DPC1(), BINARY\_FILE::READ\_BINARY\_FILE\_INTG(), BINARY\_FILE::READ\_BINARY\_FILE\_INTG1(), BINARY\_FILE::READ\_BINARY\_FILE\_LINTG(), BINARY\_FILE::READ\_BINARY\_FILE\_LINTG1(), BINARY\_FILE::READ\_BINARY\_FILE\_LOGICAL(), BINARY\_FILE::READ\_BINARY\_FILE\_LOGICAL1(), BINARY\_FILE::READ\_BINARY\_FILE\_SINTG(), BINARY\_FILE::READ\_BINARY\_FILE\_SINTG1(), BINARY\_FILE::READ\_BINARY\_FILE\_SP(), BINARY\_FILE::READ\_BINARY\_FILE\_SP1(), BINARY\_FILE::READ\_BINARY\_FILE\_SPC(), and BINARY\_FILE::READ\_BINARY\_FILE\_SPC1().

#### **6.32.2.11 INTEGER(INTG),parameter MACHINE\_CONSTANTS::MACHINE\_INT\_FORMAT = TWOS\_COMPLEMENT\_INTEGER**

Definition at line 58 of file machine\_constants\_aix.f90.

Referenced by BINARY\_FILE::OPEN\_CMISS\_BINARY\_FILE().

#### **6.32.2.12 INTEGER(INTG),parameter MACHINE\_CONSTANTS::MACHINE\_OS = AIX\_OS**

Definition at line 55 of file machine\_constants\_aix.f90.

Referenced by BASE\_ROUTINES::BASE\_ROUTINES\_INITIALISE(), BINARY\_FILE::OPEN\_CMISS\_BINARY\_FILE(), and BASE\_ROUTINES::WRITE\_STR().

#### **6.32.2.13 INTEGER(INTG),parameter MACHINE\_CONSTANTS::MACHINE\_SP\_FORMAT = SPIEEE\_NUMBER**

Definition at line 59 of file machine\_constants\_aix.f90.

Referenced by BINARY\_FILE::OPEN\_CMISS\_BINARY\_FILE().

#### **6.32.2.14 INTEGER(INTG),parameter MACHINE\_CONSTANTS::MACHINE\_TYPE = IBM\_COMPUTER**

Definition at line 54 of file machine\_constants\_aix.f90.

Referenced by BINARY\_FILE::OPEN\_CMISS\_BINARY\_FILE().

#### **6.32.2.15 INTEGER(INTG),parameter MACHINE\_CONSTANTS::SHORT\_INTEGER\_SIZE = 2**

Definition at line 62 of file machine\_constants\_aix.f90.

Referenced by BINARY\_FILE::OPEN\_CMISS\_BINARY\_FILE().

#### **6.32.2.16 INTEGER(INTG),parameter MACHINE\_CONSTANTS::SINGLE\_COMPLEX\_SIZE = 8**

Definition at line 68 of file machine\_constants\_aix.f90.

Referenced by BINARY\_FILE::OPEN\_CMISS\_BINARY\_FILE().

**6.32.2.17 INTEGER(INTG),parameter MACHINE\_CONSTANTS::SINGLE\_REAL\_SIZE = 4**

Definition at line 64 of file machine\_constants\_aix.f90.

Referenced by BINARY\_FILE::OPEN\_CMISS\_BINARY\_FILE(), BINARY\_FILE::RESET\_BINARY\_-NUMBER\_TAGS(), and BINARY\_FILE::SKIP\_CM\_BINARY\_HEADER().

## 6.33 MATHS Namespace Reference

This module contains all mathematics support routines.

### Classes

- interface [CROSS\\_PRODUCT](#)
- interface [D\\_CROSS\\_PRODUCT](#)
- interface [DETERMINANT](#)
- interface [EDP](#)
- interface [EIGENVALUE](#)
- interface [EIGENVECTOR](#)
- interface [I0](#)
- interface [I1](#)
- interface [INVERT](#)
- interface [K0](#)
- interface [K1](#)
- interface [L2NORM](#)
- interface [MATRIX\\_PRODUCT](#)
- interface [MATRIX\\_TRANSPOSE](#)
- interface [NORMALISE](#)
- interface [SOLVE\\_SMALL\\_LINEAR\\_SYSTEM](#)

### Functions

- subroutine [CROSS\\_PRODUCT\\_INTG](#) (A, B, C, ERR, ERROR,\*)
- subroutine [CROSS\\_PRODUCT\\_SP](#) (A, B, C, ERR, ERROR,\*)
- subroutine [CROSS\\_PRODUCT\\_DP](#) (A, B, C, ERR, ERROR,\*)
- subroutine [D\\_CROSS\\_PRODUCT\\_INTG](#) (N, A, B, C, D\_A, D\_B, D\_C, ERR, ERROR,\*)
- subroutine [D\\_CROSS\\_PRODUCT\\_SP](#) (N, A, B, C, D\_A, D\_B, D\_C, ERR, ERROR,\*)
- subroutine [D\\_CROSS\\_PRODUCT\\_DP](#) (N, A, B, C, D\_A, D\_B, D\_C, ERR, ERROR,\*)
- INTEGER(INTG) [DETERMINANT\\_FULL\\_INTG](#) (A, ERR, ERROR)
- REAL(SP) [DETERMINANT\\_FULL\\_SP](#) (A, ERR, ERROR)
- REAL(DP) [DETERMINANT\\_FULL\\_DP](#) (A, ERR, ERROR)
- REAL(DP) [EDP\\_DP](#) (X)
- REAL(SP) [EDP\\_SP](#) (X)
- subroutine [EIGENVALUE\\_FULL\\_SP](#) (A, EVALUES, ERR, ERROR,\*)
- subroutine [EIGENVALUE\\_FULL\\_DP](#) (A, EVALUES, ERR, ERROR,\*)
- subroutine [EIGENVECTOR\\_FULL\\_SP](#) (A, EVALUE, EVECTOR, ERR, ERROR,\*)
- subroutine [EIGENVECTOR\\_FULL\\_DP](#) (A, EVALUE, EVECTOR, ERR, ERROR,\*)
- REAL(DP) [I0\\_DP](#) (X)
- REAL(SP) [I0\\_SP](#) (X)
- REAL(DP) [I1\\_DP](#) (X)
- REAL(SP) [I1\\_SP](#) (X)
- subroutine [INVERT\\_FULL\\_SP](#) (A, B, DET, ERR, ERROR,\*)
- subroutine [INVERT\\_FULL\\_DP](#) (A, B, DET, ERR, ERROR,\*)
- REAL(DP) [K0\\_DP](#) (X)
- REAL(SP) [K0\\_SP](#) (X)
- REAL(DP) [K1\\_DP](#) (X)

- REAL(SP) [K1\\_SP](#) (X)
- REAL(DP) [KDP\\_DP](#) (X)
- REAL(SP) [KDP\\_SP](#) (X)
- REAL(SP) [L2NORM\\_SP](#) (A)
- REAL(DP) [L2NORM\\_DP](#) (A)
- subroutine [MATRIX\\_PRODUCT\\_SP](#) (A, B, C, ERR, ERROR,\*)
- subroutine [MATRIX\\_PRODUCT\\_DP](#) (A, B, C, ERR, ERROR,\*)
- subroutine [MATRIX\\_TRANSPOSE\\_SP](#) (A, AT, ERR, ERROR,\*)
- subroutine [MATRIX\\_TRANSPOSE\\_DP](#) (A, AT, ERR, ERROR,\*)
- REAL(SP) [NORMALISE\\_SP](#) (A, ERR, ERROR)
- REAL(DP) [NORMALISE\\_DP](#) (A, ERR, ERROR)
- subroutine [SOLVE\\_SMALL\\_LINEAR\\_SYSTEM\\_SP](#) (A, x, b, ERR, ERROR,\*)
- subroutine [SOLVE\\_SMALL\\_LINEAR\\_SYSTEM\\_DP](#) (A, x, b, ERR, ERROR,\*)

### 6.33.1 Detailed Description

This module contains all mathematics support routines.

### 6.33.2 Function Documentation

**6.33.2.1 subroutine MATHS::CROSS\_PRODUCT\_DP (REAL(DP),dimension(:),intent(in) A, REAL(DP),dimension(:),intent(in) B, REAL(DP),dimension(:),intent(out) C, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 250 of file maths.f90.

References [BASE\\_ROUTINES::ENTERS\(\)](#), [BASE\\_ROUTINES::ERRORS\(\)](#), and [BASE\\_ROUTINES::EXITS\(\)](#).

Here is the call graph for this function:

**6.33.2.2 subroutine MATHS::CROSS\_PRODUCT\_INTG (INTEGER(INTG),dimension(:),intent(in) A, INTEGER(INTG),dimension(:),intent(in) B, INTEGER(INTG),dimension(:),intent(out) C, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 162 of file maths.f90.

References [BASE\\_ROUTINES::ENTERS\(\)](#), [BASE\\_ROUTINES::ERRORS\(\)](#), and [BASE\\_ROUTINES::EXITS\(\)](#).

Here is the call graph for this function:

**6.33.2.3 subroutine MATHS::CROSS\_PRODUCT\_SP (REAL(SP),dimension(:),intent(in) A, REAL(SP),dimension(:),intent(in) B, REAL(SP),dimension(:),intent(out) C, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 206 of file maths.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.33.2.4 subroutine MATHS::D\_CROSS\_PRODUCT\_DP** (`INTEGER(INTG),intent(in) N`, `REAL(DP),dimension(:,),intent(in) A`, `REAL(DP),dimension(:,),intent(in) B`, `REAL(DP),dimension(:,),intent(out) C`, `REAL(DP),dimension(:,;),intent(in) D_A`, `REAL(DP),dimension(:,;),intent(in) D_B`, `REAL(DP),dimension(:,;),intent(out) D_C`, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, *`)  
`[private]`

Definition at line 415 of file maths.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.33.2.5 subroutine MATHS::D\_CROSS\_PRODUCT\_INTG** (`INTEGER(INTG),intent(in) N`, `INTEGER(INTG),dimension(:,),intent(in) A`, `INTEGER(INTG),dimension(:,),intent(in) B`, `INTEGER(INTG),dimension(:,),intent(out) C`, `INTEGER(INTG),dimension(:,;),intent(in) D_A`, `INTEGER(INTG),dimension(:,;),intent(in) D_B`, `INTEGER(INTG),dimension(:,;),intent(out) D_C`, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, *`)  
`[private]`

Definition at line 304 of file maths.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.33.2.6 subroutine MATHS::D\_CROSS\_PRODUCT\_SP** (`INTEGER(INTG),intent(in) N`, `REAL(SP),dimension(:,),intent(in) A`, `REAL(SP),dimension(:,),intent(in) B`, `REAL(SP),dimension(:,),intent(out) C`, `REAL(SP),dimension(:,;),intent(in) D_A`, `REAL(SP),dimension(:,;),intent(in) D_B`, `REAL(SP),dimension(:,;),intent(out) D_C`, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, *`)  
`[private]`

Definition at line 359 of file maths.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.33.2.7 REAL(DP) MATHS::DETERMINANT\_FULL\_DP** (`REAL(DP),dimension(:,;),intent(in) A`, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR`)  
`[private]`

Definition at line 572 of file maths.f90.

References `KINDS::_DP`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.33.2.8 INTEGER(INTG) MATHS::DETERMINANT\_FULL\_INTG  
(INTEGER(INTG),dimension(:,:),intent(in) A, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

Definition at line 480 of file maths.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.33.2.9 REAL(SP) MATHS::DETERMINANT\_FULL\_SP (REAL(SP),dimension(:,:),intent(in)  
A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR)  
[private]**

Definition at line 526 of file maths.f90.

References KINDS::\_SP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.33.2.10 REAL(DP) MATHS::EDP\_DP (REAL(DP),intent(in) X) [private]**

Definition at line 627 of file maths.f90.

References KINDS::\_DP.

**6.33.2.11 REAL(SP) MATHS::EDP\_SP (REAL(SP),intent(in) X) [private]**

Definition at line 663 of file maths.f90.

References KINDS::\_SP.

**6.33.2.12 subroutine MATHS::EIGENVALUE\_FULL\_DP (REAL(DP),dimension(:,:),intent(in) A,  
REAL(DP),dimension(:,intent(out) EVALUES, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 800 of file maths.f90.

References KINDS::\_DP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.33.2.13 subroutine MATHS::EIGENVALUE\_FULL\_SP (REAL(SP),dimension(:,:),intent(in) A,  
REAL(SP),dimension(:,intent(out) EVALUES, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 708 of file maths.f90.

References KINDS::\_SP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.33.2.14 subroutine MATHS::EIGENVECTOR\_FULL\_DP (REAL(DP),dimension(:,:),intent(in) A, REAL(DP),intent(in) EVALUE, REAL(DP),dimension(:,intent(out) EVECTOR, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 992 of file maths.f90.

References KINDS::\_DP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.33.2.15 subroutine MATHS::EIGENVECTOR\_FULL\_SP (REAL(SP),dimension(:,:),intent(in) A, REAL(SP),intent(in) EVALUE, REAL(SP),dimension(:,intent(out) EVECTOR, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 901 of file maths.f90.

References KINDS::\_DP, KINDS::\_SP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.33.2.16 REAL(DP) MATHS::I0\_DP (REAL(DP),intent(in) X) [private]**

Definition at line 1092 of file maths.f90.

References KINDS::\_DP.

**6.33.2.17 REAL(SP) MATHS::I0\_SP (REAL(SP),intent(in) X) [private]**

Definition at line 1127 of file maths.f90.

References KINDS::\_SP.

**6.33.2.18 REAL(DP) MATHS::I1\_DP (REAL(DP),intent(in) X) [private]**

Definition at line 1171 of file maths.f90.

References KINDS::\_DP.

**6.33.2.19 REAL(SP) MATHS::I1\_SP (REAL(SP),intent(in) X) [private]**

Definition at line 1206 of file maths.f90.

References KINDS::\_SP.

---

**6.33.2.20 subroutine MATHS::INVERT\_FULL\_DP (REAL(DP),dimension(:,:),intent(in) A, REAL(DP),dimension(:,:),intent(out) B, REAL(DP),intent(out) DET, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 1327 of file maths.f90.

References KINDS::\_DP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.33.2.21 subroutine MATHS::INVERT\_FULL\_SP (REAL(SP),dimension(:,:),intent(in) A, REAL(SP),dimension(:,:),intent(out) B, REAL(SP),intent(out) DET, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 1250 of file maths.f90.

References KINDS::\_DP, KINDS::\_SP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.33.2.22 REAL(DP) MATHS::K0\_DP (REAL(DP),intent(in) X) [private]**

Definition at line 1413 of file maths.f90.

References KINDS::\_DP.

**6.33.2.23 REAL(SP) MATHS::K0\_SP (REAL(SP),intent(in) X) [private]**

Definition at line 1464 of file maths.f90.

References KINDS::\_SP.

**6.33.2.24 REAL(DP) MATHS::K1\_DP (REAL(DP),intent(in) X) [private]**

Definition at line 1524 of file maths.f90.

References KINDS::\_DP.

**6.33.2.25 REAL(SP) MATHS::K1\_SP (REAL(SP),intent(in) X) [private]**

Definition at line 1576 of file maths.f90.

References KINDS::\_SP.

**6.33.2.26 REAL(DP) MATHS::KDP\_DP (REAL(DP),intent(in) X) [private]**

Definition at line 1637 of file maths.f90.

References KINDS::\_DP.

**6.33.2.27 REAL(SP) MATHS::KDP\_SP (REAL(SP),intent(in) X) [private]**

Definition at line 1675 of file maths.f90.

References KINDS::\_SP.

**6.33.2.28 REAL(DP) MATHS::L2NORM\_DP (REAL(DP),dimension(:,),intent(in) A) [private]**

Definition at line 1746 of file maths.f90.

**6.33.2.29 REAL(SP) MATHS::L2NORM\_SP (REAL(SP),dimension(:,),intent(in) A) [private]**

Definition at line 1722 of file maths.f90.

**6.33.2.30 subroutine MATHS::MATRIX\_PRODUCT\_DP (REAL(DP),dimension(3,3),intent(in) A, REAL(DP),dimension(3,3),intent(in) B, REAL(DP),dimension(3,3),intent(out) C, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 1808 of file maths.f90.

References KINDS::\_DP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.33.2.31 subroutine MATHS::MATRIX\_PRODUCT\_SP (REAL(SP),dimension(3,3),intent(in) A, REAL(SP),dimension(3,3),intent(in) B, REAL(SP),dimension(3,3),intent(out) C, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 1770 of file maths.f90.

References KINDS::\_SP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.33.2.32 subroutine MATHS::MATRIX\_TRANSPOSE\_DP (REAL(DP),dimension(3,3),intent(in) A, REAL(DP),dimension(3,3),intent(out) AT, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 1881 of file maths.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

---

**6.33.2.33 subroutine MATHS::MATRIX\_TRANSPOSE\_SP (REAL(SP),dimension(3,3),intent(in) A, REAL(SP),dimension(3,3),intent(out) AT, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 1846 of file maths.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.33.2.34    REAL(DP) MATHS::NORMALISE\_DP (REAL(DP),dimension(:,),intent(in) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

Definition at line 1962 of file maths.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.33.2.35    REAL(SP) MATHS::NORMALISE\_SP (REAL(SP),dimension(:,),intent(in) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

Definition at line 1925 of file maths.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.33.2.36 subroutine MATHS::SOLVE\_SMALL\_LINEAR\_SYSTEM\_DP (REAL(DP),dimension(:,,:),intent(in) A, REAL(DP),dimension(:,),intent(out) x, REAL(DP),dimension(:,),intent(in) b, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 2057 of file maths.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.33.2.37 subroutine MATHS::SOLVE\_SMALL\_LINEAR\_SYSTEM\_SP (REAL(SP),dimension(:,,:),intent(in) A, REAL(SP),dimension(:,),intent(out) x, REAL(SP),dimension(:,),intent(in) b, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 2008 of file maths.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

## 6.34 MATRIX\_VECTOR Namespace Reference

This module contains all routines dealing with (non-distributed) matrix and vectors types.

### Classes

- interface [MATRIX\\_ALL\\_VALUES\\_SET](#)
- interface [MATRIX\\_DATA\\_GET](#)
- interface [MATRIX\\_VALUES\\_ADD](#)
- interface [MATRIX\\_VALUES\\_GET](#)
- interface [MATRIX\\_VALUES\\_SET](#)
- interface [VECTOR\\_ALL\\_VALUES\\_SET](#)
- interface [VECTOR\\_DATA\\_GET](#)
- interface [VECTOR\\_VALUES\\_GET](#)
- interface [VECTOR\\_VALUES\\_SET](#)

### Functions

- subroutine [MATRIX\\_ALL\\_VALUES\\_SET\\_INTG](#) (MATRIX, VALUE, ERR, ERROR,\*)
 

*Sets all values in an integer matrix to the specified value.*
- subroutine [MATRIX\\_ALL\\_VALUES\\_SET\\_SP](#) (MATRIX, VALUE, ERR, ERROR,\*)
 

*Sets all values in a single precision matrix to the specified value.*
- subroutine [MATRIX\\_ALL\\_VALUES\\_SET\\_DP](#) (MATRIX, VALUE, ERR, ERROR,\*)
 

*Sets all values in a double precision matrix to the specified value.*
- subroutine [MATRIX\\_ALL\\_VALUES\\_SET\\_L](#) (MATRIX, VALUE, ERR, ERROR,\*)
 

*Sets all values in a logical matrix to the specified value.*
- subroutine [MATRIX\\_CREATE\\_FINISH](#) (MATRIX, ERR, ERROR,\*)
 

*Finishes the creation a matrix.*
- subroutine [MATRIX\\_CREATE\\_START](#) (MATRIX, ERR, ERROR,\*)
 

*Starts the creation a matrix.*
- subroutine [MATRIX\\_DATA\\_GET\\_INTG](#) (MATRIX, DATA, ERR, ERROR,\*)
 

*Returns a pointer to the data of an integer matrix. Note: the values can be used for read operations but a [MATRIX\\_VALUES\\_SET](#) call must be used to change any values. The pointer should not be deallocated.*
- subroutine [MATRIX\\_DATA\\_GET\\_SP](#) (MATRIX, DATA, ERR, ERROR,\*)
 

*Returns a pointer to the data of a single precision matrix. Note: the values can be used for read operations but a [MATRIX\\_VALUES\\_SET](#) call must be used to change any values. The pointer should not be deallocated.*
- subroutine [MATRIX\\_DATA\\_GET\\_DP](#) (MATRIX, DATA, ERR, ERROR,\*)
 

*Returns a pointer to the data of a double precision matrix. Note: the values can be used for read operations but a [MATRIX\\_VALUES\\_SET](#) call must be used to change any values. The pointer should not be deallocated.*

- subroutine [MATRIX\\_DATA\\_GET\\_L](#) (MATRIX, DATA, ERR, ERROR,\*)
 

*Returns a pointer to the data of a logical matrix. Note: the values can be used for read operations but a [MATRIX\\_VALUES\\_SET](#) call must be used to change any values. The pointer should not be deallocated.*
- subroutine [MATRIX\\_DATA\\_TYPE\\_SET](#) (MATRIX, DATA\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the data type of a matrix.*
- subroutine [MATRIX\\_DESTROY](#) (MATRIX, ERR, ERROR,\*)
 

*Destroys a matrix.*
- subroutine [MATRIX\\_DUPLICATE](#) (MATRIX, NEW\_MATRIX, ERR, ERROR,\*)
 

*Duplicates the matrix and returns a pointer to the duplicated matrix in NEWMATRIX.*
- subroutine [MATRIX\\_FINALISE](#) (MATRIX, ERR, ERROR,\*)
 

*Finalises a matrix and deallocates all memory.*
- subroutine [MATRIX\\_INITIALISE](#) (MATRIX, ERR, ERROR,\*)
 

*Initialises a matrix.*
- subroutine [MATRIX\\_MAX\\_COLUMNS\\_PER\\_ROW\\_GET](#) (MATRIX, MAX\_COLUMNS\_PER\_ROW, ERR, ERROR,\*)
 

*Gets the maximum number of columns in each row of a distributed matrix.*
- subroutine [MATRIX\\_NUMBER\\_NON\\_ZEROS\\_SET](#) (MATRIX, NUMBER\_NON\_ZEROS, ERR, ERROR,\*)
 

*Sets/changes the number of non zeros for a matrix.*
- subroutine [MATRIX\\_MAX\\_SIZE\\_SET](#) (MATRIX, MAX\_M, MAX\_N, ERR, ERROR,\*)
 

*Sets/changes the maximum size of a matrix.*
- subroutine [MATRIX\\_OUTPUT](#) (ID, MATRIX, ERR, ERROR,\*)
 

*Sets/changes the size of a matrix.*
- subroutine [MATRIX\\_SIZE\\_SET](#) (MATRIX, M, N, ERR, ERROR,\*)
 

*Sets/changes the size of a matrix.*
- subroutine [MATRIX\\_STORAGE\\_LOCATION\\_FIND](#) (MATRIX, I, J, LOCATION, ERR, ERROR,\*)
 

*Returns the storage location in the data array of a matrix that corresponds to location I,J. If the location does not exist the routine returns zero.*
- subroutine [MATRIX\\_STORAGE\\_LOCATIONS\\_GET](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, ERR, ERROR,\*)
 

*Gets the storage locations (sparsity pattern) of a matrix.*
- subroutine [MATRIX\\_STORAGE\\_LOCATIONS\\_SET](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, ERR, ERROR,\*)
 

*Sets the storage locations (sparsity pattern) in a matrix to that specified by the row and column indices.*
- subroutine [MATRIX\\_STORAGE\\_TYPE\\_GET](#) (MATRIX, STORAGE\_TYPE, ERR, ERROR,\*)

*Gets the storage type for a matrix.*

- subroutine [MATRIX\\_STORAGE\\_TYPE\\_SET](#) (MATRIX, STORAGE\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the storage type for a matrix.*
- subroutine [MATRIX\\_VALUES\\_ADD\\_INTG](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Adds values to an integer matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.*
- subroutine [MATRIX\\_VALUES\\_ADD\\_INTG1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
 

*Adds a value to an integer matrix at the location specified by the row and column index i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.*
- subroutine [MATRIX\\_VALUES\\_ADD\\_INTG2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Adds a matrix of values to an integer matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.*
- subroutine [MATRIX\\_VALUES\\_ADD\\_SP](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Adds values to a single precision real matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.*
- subroutine [MATRIX\\_VALUES\\_ADD\\_SP1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
 

*Adds a value to a single precision real matrix at the location specified by the row and column index i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.*
- subroutine [MATRIX\\_VALUES\\_ADD\\_SP2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Adds a matrix of values to a single precision real matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.*
- subroutine [MATRIX\\_VALUES\\_ADD\\_DP](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Adds values to a double precision real matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.*
- subroutine [MATRIX\\_VALUES\\_ADD\\_DP1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
 

*Adds a value to a double precision real matrix at the location specified by the row and column index i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.*
- subroutine [MATRIX\\_VALUES\\_ADD\\_DP2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Adds a matrix of values to a double precision real matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.*
- subroutine [MATRIX\\_VALUES\\_ADD\\_L](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Adds values to a logical matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.*

*Adds values to a logical matrix at the location specified by the row and column indices i.e.,  $MATRIX(I,J)=MATRIX(I,J).OR.VALUE$ .*

- subroutine [MATRIX\\_VALUES\\_ADD\\_L1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)

*Adds a value to a logical matrix at the location specified by the row and column index i.e.,  $MATRIX(I,J)=MATRIX(I,J).OR.VALUE$ .*

- subroutine [MATRIX\\_VALUES\\_ADD\\_L2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)

*Adds a matrix of values to a logical matrix at the location specified by the row and column indices i.e.,  $MATRIX(I,J)=MATRIX(I,J).OR.VALUE$ .*

- subroutine [MATRIX\\_VALUES\\_GET\\_INTG](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)

*Gets the values in an integer matrix at the location specified by the row and column indices i.e.,  $VALUE=MATRIX(I,J)$ .*

- subroutine [MATRIX\\_VALUES\\_GET\\_INTG1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)

*Gets a value in an integer matrix at the location specified by the row and column index i.e.,  $VALUE=MATRIX(I,J)$ .*

- subroutine [MATRIX\\_VALUES\\_GET\\_INTG2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)

*Gets the matrix of values in an integer matrix at the location specified by the row and column indices i.e.,  $VALUE=MATRIX(I,J)$ .*

- subroutine [MATRIX\\_VALUES\\_GET\\_SP](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)

*Gets values in a single precision real matrix at the location specified by the row and column indices i.e.,  $VALUE=MATRIX(I,J)$ .*

- subroutine [MATRIX\\_VALUES\\_GET\\_SP1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)

*Gets a value in a single precision real matrix at the location specified by the row and column index i.e.,  $VALUE=MATRIX(I,J)$ .*

- subroutine [MATRIX\\_VALUES\\_GET\\_SP2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)

*Gets a matrix of values in a single precision real matrix at the location specified by the row and column indices i.e.,  $VALUE=MATRIX(I,J)$ .*

- subroutine [MATRIX\\_VALUES\\_GET\\_DP](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)

*Gets values in a double precision real matrix at the location specified by the row and column indices i.e.,  $VALUE=MATRIX(I,J)$ .*

- subroutine [MATRIX\\_VALUES\\_GET\\_DP1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)

*Gets a value in a double precision real matrix at the location specified by the row and column index i.e.,  $VALUE=MATRIX(I,J)$ .*

- subroutine [MATRIX\\_VALUES\\_GET\\_DP2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Gets a matrix of values in a double precision real matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).*
- subroutine [MATRIX\\_VALUES\\_GET\\_L](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Gets values in a logical matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).*
- subroutine [MATRIX\\_VALUES\\_GET\\_L1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
 

*Gets a value in a logical matrix at the location specified by the row and column index i.e., VALUE=MATRIX(I,J).*
- subroutine [MATRIX\\_VALUES\\_GET\\_L2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Gets a matrix of values in a logical matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).*
- subroutine [MATRIX\\_VALUES\\_SET\\_INTG](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Sets the values in an integer matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=VALUE.*
- subroutine [MATRIX\\_VALUES\\_SET\\_INTG1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
 

*Sets a value in an integer matrix at the location specified by the row and column index i.e., MATRIX(I,J)=VALUE.*
- subroutine [MATRIX\\_VALUES\\_SET\\_INTG2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Sets the matrix of values in an integer matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=VALUE.*
- subroutine [MATRIX\\_VALUES\\_SET\\_SP](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Sets the values in a single precision real matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=VALUE.*
- subroutine [MATRIX\\_VALUES\\_SET\\_SP1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
 

*Sets the value in a single precision real matrix at the location specified by the row and column index i.e., MATRIX(I,J)=VALUE.*
- subroutine [MATRIX\\_VALUES\\_SET\\_SP2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Sets the matrix of values in a single precision real matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=VALUE.*

- subroutine [MATRIX\\_VALUES\\_SET\\_DP](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Sets the values in a double precision real matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I,J)=\text{VALUE}$ .*
- subroutine [MATRIX\\_VALUES\\_SET\\_DPI](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
 

*Sets a value in a double precision real matrix at the location specified by the row and column index i.e.,  $\text{MATRIX}(I,J)=\text{VALUE}$ .*
- subroutine [MATRIX\\_VALUES\\_SET\\_DP2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Sets the matrix of values in a double precision real matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I,J)=\text{VALUE}$ .*
- subroutine [MATRIX\\_VALUES\\_SET\\_L](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Sets the values in a logical matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I,J)=\text{VALUE}$ .*
- subroutine [MATRIX\\_VALUES\\_SET\\_L1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
 

*Sets a value in a logical matrix at the location specified by the row and column index i.e.,  $\text{MATRIX}(I,J)=\text{VALUE}$ .*
- subroutine [MATRIX\\_VALUES\\_SET\\_L2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Sets the matrix of values in a logical matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I,J)=\text{VALUE}$ .*
- subroutine [VECTOR\\_ALL\\_VALUES\\_SET\\_INTG](#) (VECTOR, VALUE, ERR, ERROR,\*)
 

*Sets all values in an integer vector to the specified value.*
- subroutine [VECTOR\\_ALL\\_VALUES\\_SET\\_SP](#) (VECTOR, VALUE, ERR, ERROR,\*)
 

*Sets all values in a single precision vector to the specified value.*
- subroutine [VECTOR\\_ALL\\_VALUES\\_SET\\_DP](#) (VECTOR, VALUE, ERR, ERROR,\*)
 

*Sets all values in a double precision vector to the specified value.*
- subroutine [VECTOR\\_ALL\\_VALUES\\_SET\\_L](#) (VECTOR, VALUE, ERR, ERROR,\*)
 

*Sets all values in a logical vector to the specified value.*
- subroutine [VECTOR\\_CREATE\\_FINISH](#) (VECTOR, ERR, ERROR,\*)
 

*Finishes the creation of a vector.*
- subroutine [VECTOR\\_CREATE\\_START](#) (VECTOR, ERR, ERROR,\*)
 

*Starts the creation a vector.*
- subroutine [VECTOR\\_DATA\\_GET\\_INTG](#) (VECTOR, DATA, ERR, ERROR,\*)
 

*Returns a pointer to the data of an integer vector. Note: the values can be used for read operations but a [VECTOR\\_VALUES\\_SET](#) call must be used to change any values. The pointer should not be deallocated.*

- subroutine [VECTOR\\_DATA\\_GET\\_SP](#) (VECTOR, DATA, ERR, ERROR,\*)
 

*Returns a pointer to the data of a single precision vector. Note: the values can be used for read operations but a [VECTOR\\_VALUES\\_SET](#) call must be used to change any values. The pointer should not be deallocated.*
- subroutine [VECTOR\\_DATA\\_GET\\_DP](#) (VECTOR, DATA, ERR, ERROR,\*)
 

*Returns a pointer to the data of a double precision vector. Note: the values can be used for read operations but a [VECTOR\\_VALUES\\_SET](#) call must be used to change any values. The pointer should not be deallocated.*
- subroutine [VECTOR\\_DATA\\_GET\\_L](#) (VECTOR, DATA, ERR, ERROR,\*)
 

*Returns a pointer to the data of a logical vector. Note: the values can be used for read operations but a [VECTOR\\_VALUES\\_SET](#) call must be used to change any values. The pointer should not be deallocated.*
- subroutine [VECTOR\\_DATA\\_TYPE\\_SET](#) (VECTOR, DATA\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the data type of a vector.*
- subroutine [VECTOR\\_DESTROY](#) (VECTOR, ERR, ERROR,\*)
 

*Destroys a vector.*
- subroutine [VECTOR\\_DUPLICATE](#) (VECTOR, NEW\_VECTOR, ERR, ERROR,\*)
 

*Duplicates a vector structure and returns a pointer to the new vector in NEW\_VECTOR.*
- subroutine [VECTOR\\_FINALISE](#) (VECTOR, ERR, ERROR,\*)
 

*Finalises a vector and deallocates all memory.*
- subroutine [VECTOR\\_INITIALISE](#) (VECTOR, ERR, ERROR,\*)
 

*Initialises a vector.*
- subroutine [VECTOR\\_SIZE\\_SET](#) (VECTOR, N, ERR, ERROR,\*)
 

*Sets/changes the size of a vector.*
- subroutine [VECTOR\\_VALUES\\_GET\\_INTG](#) (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
 

*Gets the values in an integer vector at the indices specified.*
- subroutine [VECTOR\\_VALUES\\_GET\\_INTG1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
 

*Gets a value in an integer vector at the location specified by the index.*
- subroutine [VECTOR\\_VALUES\\_GET\\_SP](#) (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
 

*Gets the values in a single precision real vector at the indices specified.*
- subroutine [VECTOR\\_VALUES\\_GET\\_SP1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
 

*Gets a value in a single precision vector at the location specified by the index.*
- subroutine [VECTOR\\_VALUES\\_GET\\_DP](#) (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
 

*Gets the values in a double precision real vector at the indices specified.*
- subroutine [VECTOR\\_VALUES\\_GET\\_DP1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
 

*Gets a value in a double precision vector at the location specified by the index.*

- subroutine [VECTOR\\_VALUES\\_GET\\_L](#) (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
 

*Gets the values in a logical real vector at the indices specified.*
- subroutine [VECTOR\\_VALUES\\_GET\\_L1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
 

*Gets a value in a logical vector at the location specified by the index.*
- subroutine [VECTOR\\_VALUES\\_SET\\_INTG](#) (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
 

*Sets the values in an integer vector at the specified indices.*
- subroutine [VECTOR\\_VALUES\\_SET\\_INTG1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
 

*Sets a value in an integer vector at the specified index.*
- subroutine [VECTOR\\_VALUES\\_SET\\_SP](#) (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
 

*Sets the values in a single precision vector at the specified indices.*
- subroutine [VECTOR\\_VALUES\\_SET\\_SP1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
 

*Sets a value in a single precision vector at the specified index.*
- subroutine [VECTOR\\_VALUES\\_SET\\_DP](#) (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
 

*Sets the values in a double precision vector at the specified indices.*
- subroutine [VECTOR\\_VALUES\\_SET\\_DP1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
 

*Sets a value in a double precision vector at the specified index.*
- subroutine [VECTOR\\_VALUES\\_SET\\_L](#) (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
 

*Sets the values in a logical vector at the specified indices.*
- subroutine [VECTOR\\_VALUES\\_SET\\_L1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
 

*Sets a value in a logical vector at the specified index.*

## Variables

- INTEGER(INTG), parameter [MATRIX\\_VECTOR\\_INTG\\_TYPE](#) = INTEGER\_TYPE
 

*Integer matrix-vector data type.*
- INTEGER(INTG), parameter [MATRIX\\_VECTOR\\_SP\\_TYPE](#) = SINGLE\_REAL\_TYPE
 

*Single precision real matrix-vector data type.*
- INTEGER(INTG), parameter [MATRIX\\_VECTOR\\_DP\\_TYPE](#) = DOUBLE\_REAL\_TYPE
 

*Double precision real matrix-vector data type.*
- INTEGER(INTG), parameter [MATRIX\\_VECTOR\\_L\\_TYPE](#) = LOGICAL\_TYPE
 

*Logical matrix-vector data type.*
- INTEGER(INTG), parameter [MATRIX\\_BLOCK\\_STORAGE\\_TYPE](#) = 0
 

*Matrix block storage type.*
- INTEGER(INTG), parameter [MATRIX\\_DIAGONAL\\_STORAGE\\_TYPE](#) = 1

*Matrix diagonal storage type.*

- INTEGER(INTG), parameter **MATRIX\_COLUMN\_MAJOR\_STORAGE\_TYPE** = 2  
*Matrix column major storage type.*
- INTEGER(INTG), parameter **MATRIX\_ROW\_MAJOR\_STORAGE\_TYPE** = 3  
*Matrix row major storage type.*
- INTEGER(INTG), parameter **MATRIX\_COMPRESSED\_ROW\_STORAGE\_TYPE** = 4  
*Matrix compressed row storage type.*
- INTEGER(INTG), parameter **MATRIX\_COMPRESSED\_COLUMN\_STORAGE\_TYPE** = 5  
*Matrix compressed column storage type.*
- INTEGER(INTG), parameter **MATRIX\_ROW\_COLUMN\_STORAGE\_TYPE** = 6  
*Matrix row-column storage type.*
- INTEGER(INTG), save **MATRIX\_VECTOR\_ID** = 1

### 6.34.1 Detailed Description

This module contains all routines dealing with (non-distributed) matrix and vectors types.

### 6.34.2 Function Documentation

#### 6.34.2.1 subroutine MATRIX\_VECTOR::MATRIX\_ALL\_VALUES\_SET\_DP (TYPE(MATRIX\_TYPE),pointer **MATRIX**, REAL(DP),intent(in) **VALUE**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

Sets all values in a double precision matrix to the specified value.

##### Parameters:

**MATRIX** A pointer to the matrix

**VALUE** The value to set

**ERR** The error code

**ERROR** The error string

Definition at line 373 of file matrix\_vector.f90.

References      **BASE\_ROUTINES::ENTERS()**,      **BASE\_ROUTINES::ERRORS()**,      **BASE\_ROUTINES::EXITS()**, and **MATRIX\_VECTOR\_DP\_TYPE**.

Here is the call graph for this function:

#### 6.34.2.2 subroutine MATRIX\_VECTOR::MATRIX\_ALL\_VALUES\_SET\_INTG (TYPE(MATRIX\_TYPE),pointer **MATRIX**, INTEGER(INTG),intent(in) **VALUE**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

Sets all values in an integer matrix to the specified value.

**Parameters:**

**MATRIX** A pointer to the matrix  
**VALUE** The value to set  
**ERR** The error code  
**ERROR** The error string

Definition at line 293 of file matrix\_vector.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), and MATRIX\_VECTOR\_INTG\_TYPE.

Here is the call graph for this function:

**6.34.2.3 subroutine MATRIX\_VECTOR::MATRIX\_ALL\_VALUES\_SET\_L**  
**(TYPE(MATRIX\_TYPE),pointer MATRIX, LOGICAL,intent(in) VALUE,**  
**INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets all values in a logical matrix to the specified value.

**Parameters:**

**MATRIX** A pointer to the matrix  
**VALUE** The value to set  
**ERR** The error code  
**ERROR** The error string

Definition at line 413 of file matrix\_vector.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), and MATRIX\_VECTOR\_L\_TYPE.

Here is the call graph for this function:

**6.34.2.4 subroutine MATRIX\_VECTOR::MATRIX\_ALL\_VALUES\_SET\_SP**  
**(TYPE(MATRIX\_TYPE),pointer MATRIX, REAL(SP),intent(in) VALUE,**  
**INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets all values in a single precision matrix to the specified value.

**Parameters:**

**MATRIX** A pointer to the matrix  
**VALUE** The value to set  
**ERR** The error code  
**ERROR** The error string

Definition at line 333 of file matrix\_vector.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), and MATRIX\_VECTOR\_SP\_TYPE.

Here is the call graph for this function:

**6.34.2.5 subroutine MATRIX\_VECTOR::MATRIX\_CREATE\_FINISH**  
 (TYPE(MATRIX\_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Finishes the creation a matrix.

**Parameters:**

*MATRIX* A pointer to the matrix to finish

*ERR* The error code

*ERROR* The error string

Definition at line 453 of file matrix\_vector.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(),    MATRIX\_BLOCK\_STORAGE\_TYPE,    MATRIX\_COLUMN\_MAJOR\_-STORAGE\_TYPE,    MATRIX\_COMPRESSED\_COLUMN\_STORAGE\_TYPE,    MATRIX\_-COMPRESSED\_ROW\_STORAGE\_TYPE,    MATRIX\_DIAGONAL\_STORAGE\_TYPE,    MATRIX\_-ROW\_COLUMN\_STORAGE\_TYPE,    MATRIX\_ROW\_MAJOR\_STORAGE\_TYPE,    MATRIX\_-VECTOR\_DP\_TYPE,    MATRIX\_VECTOR\_ID,    MATRIX\_VECTOR\_INTG\_TYPE,    MATRIX\_-VECTOR\_L\_TYPE, and MATRIX\_VECTOR\_SP\_TYPE.

Referenced by MATRIX\_DUPLICATE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.34.2.6 subroutine MATRIX\_VECTOR::MATRIX\_CREATE\_START**  
 (TYPE(MATRIX\_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Starts the creation a matrix.

**Parameters:**

*MATRIX* A pointer to the matrix

*ERR* The error code

*ERROR* The error string

Definition at line 588 of file matrix\_vector.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(),    MATRIX\_BLOCK\_STORAGE\_TYPE,    MATRIX\_FINALISE(),    MATRIX\_-INITIALISE(), and MATRIX\_VECTOR\_DP\_TYPE.

Referenced by MATRIX\_DUPLICATE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.34.2.7 subroutine MATRIX\_VECTOR::MATRIX\_DATA\_GET\_DP** (TYPE(MATRIX\_-TYPE),pointer *MATRIX*, REAL(DP),dimension(:),pointer *DATA*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Returns a pointer to the data of a double precision matrix. Note: the values can be used for read operations but a [MATRIX\\_VALUES\\_SET](#) call must be used to change any values. The pointer should not be

deallocated.

**Parameters:**

- MATRIX** A pointer to the matrix
- DATA** On return a pointer to the matrix data
- ERR** The error code
- ERROR** The error string

Definition at line 712 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(), and MATRIX\_VECTOR\_DP\_TYPE.

Here is the call graph for this function:

**6.34.2.8 subroutine MATRIX\_VECTOR::MATRIX\_DATA\_GET\_INTG (TYPE(MATRIX\_-TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),pointer DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Returns a pointer to the data of an integer matrix. Note: the values can be used for read operations but a [MATRIX\\_VALUES\\_SET](#) call must be used to change any values. The pointer should not be deallocated.

**Parameters:**

- MATRIX** A pointer to the matrix
- DATA** On return a pointer to the matrix data
- ERR** The error code
- ERROR** The error string

Definition at line 622 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(), and MATRIX\_VECTOR\_INTG\_TYPE.

Here is the call graph for this function:

**6.34.2.9 subroutine MATRIX\_VECTOR::MATRIX\_DATA\_GET\_L (TYPE(MATRIX\_-TYPE),pointer MATRIX, LOGICAL,dimension(:),pointer DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Returns a pointer to the data of a logical matrix. Note: the values can be used for read operations but a [MATRIX\\_VALUES\\_SET](#) call must be used to change any values. The pointer should not be deallocated.

**Parameters:**

- MATRIX** A pointer to the matrix
- DATA** On return a pointer to the matrix data
- ERR** The error code
- ERROR** The error string

Definition at line 757 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(), and MATRIX\_VECTOR\_L\_TYPE.

Here is the call graph for this function:

---

**6.34.2.10 subroutine MATRIX\_VECTOR::MATRIX\_DATA\_GET\_SP (TYPE(MATRIX\_TYPE),pointer *MATRIX*, REAL(SP),dimension(:),pointer *DATA*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Returns a pointer to the data of a single precision matrix. Note: the values can be used for read operations but aMATRIX\_VALUES\_SET call must be used to change any values. The pointer should not be deallocated.

**Parameters:**

***MATRIX*** A pointer to the matrix  
***DATA*** On return a pointer to the matrix data  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 667 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_ROUTINES::EXITS(), and MATRIX\_VECTOR\_SP\_TYPE.

Here is the call graph for this function:

**6.34.2.11 subroutine MATRIX\_VECTOR::MATRIX\_DATA\_TYPE\_SET (TYPE(MATRIX\_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(in) *DATA\_TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Sets/changes the data type of a matrix.

**Parameters:**

***MATRIX*** A pointer to the matrix  
***DATA\_TYPE*** The data type to set for the matrix.

**See also:**

[MATRIX\\_VECTOR::DataTypes](#), [MATRIX\\_VECTOR](#)

***ERR*** The error code

***ERROR*** The error string

Definition at line 802 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_ROUTINES::EXITS(), MATRIX\_VECTOR\_DP\_TYPE, MATRIX\_VECTOR\_INTG\_TYPE, MATRIX\_VECTOR\_L\_TYPE, and MATRIX\_VECTOR\_SP\_TYPE.

Referenced by MATRIX\_DUPLICATE().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.34.2.12 subroutine MATRIX\_VECTOR::MATRIX\_DESTROY (TYPE(MATRIX\_TYPE),pointer MATRIX, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Destroys a matrix.

**Parameters:**

*MATRIX* A pointer to the matrix to destroy

*ERR* The error code

*ERROR* The error string

Definition at line 848 of file matrix\_vector.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_ROUTINES::EXITS(), and MATRIX\_FINALISE().

Here is the call graph for this function:

**6.34.2.13 subroutine MATRIX\_VECTOR::MATRIX\_DUPLICATE (TYPE(MATRIX\_TYPE),pointer MATRIX, TYPE(MATRIX\_TYPE),pointer NEW\_MATRIX, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Duplicates the matrix and returns a pointer to the duplicated matrix in NEWMATRIX.

**Parameters:**

*MATRIX* A pointer to the matrix to duplicate

*NEW\_MATRIX* On return a pointer to a new duplicated matrix

*ERR* The error code

*ERROR* The error string

Definition at line 875 of file matrix\_vector.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_ROUTINES::EXITS(),      MATRIX\_BLOCK\_STORAGE\_TYPE,      MATRIX\_COLUMN\_MAJOR\_STORAGE\_TYPE,      MATRIX\_COMPRESSED\_COLUMN\_STORAGE\_TYPE,      MATRIX\_COMPRESSED\_ROW\_STORAGE\_TYPE,      MATRIX\_CREATE\_FINISH(),      MATRIX\_CREATE\_START(),      MATRIX\_DATA\_TYPE\_SET(),      MATRIX\_DIAGONAL\_STORAGE\_TYPE,      MATRIX\_FINALISE(),      MATRIX\_MAX\_SIZE\_SET(),      MATRIX\_NUMBER\_NON\_ZEROS\_SET(),      MATRIX\_ROW\_COLUMN\_STORAGE\_TYPE,      MATRIX\_ROW\_MAJOR\_STORAGE\_TYPE,      MATRIX\_SIZE\_SET(),      MATRIX\_STORAGE\_LOCATIONS\_SET(), and MATRIX\_STORAGE\_TYPE\_SET().

Here is the call graph for this function:

**6.34.2.14 subroutine MATRIX\_VECTOR::MATRIX\_FINALISE (TYPE(MATRIX\_TYPE),pointer MATRIX, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Finalises a matrix and deallocates all memory.

**Parameters:**

*MATRIX* A pointer to the matrix to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 926 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `MATRIX_CREATE_START()`, `MATRIX_DESTROY()`, and `MATRIX_DUPLICATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.34.2.15 subroutine `MATRIX_VECTOR::MATRIX_INITIALISE` (`TYPE(MATRIX_TYPE)`,pointer *MATRIX*, `INTEGER(INTG)`,`intent(out)` *ERR*, `TYPE(VARYING_STRING)`,`intent(out)` *ERROR*, \*)

Initialises a matrix.

##### Parameters:

**MATRIX** A pointer to the matrix

**ERR** The error code

**ERROR** The error string

Definition at line 958 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `MATRIX_CREATE_START()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.34.2.16 subroutine `MATRIX_VECTOR::MATRIX_MAX_COLUMNS_PER_ROW_GET` (`TYPE(MATRIX_TYPE)`,pointer *MATRIX*, `INTEGER(INTG)`,`intent(out)` *MAX\_COLUMNS\_PER\_ROW*, `INTEGER(INTG)`,`intent(out)` *ERR*, `TYPE(VARYING_STRING)`,`intent(out)` *ERROR*, \*)

Gets the maximum number of columns in each row of a distributed matrix.

##### Parameters:

**MATRIX** A pointer to the matrix

**MAX\_COLUMNS\_PER\_ROW** On return, the maximum number of columns in each row

**ERR** The error code

**ERROR** The error string

Definition at line 997 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.34.2.17 subroutine MATRIX\_VECTOR::MATRIX\_MAX\_SIZE\_SET**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) MAX_M,  
 INTEGER(INTG),intent(in) MAX_N, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets/changes the maximum size of a matrix.

**Parameters:**

***MATRIX*** A pointer to the matrix  
***MAX\_M*** The maximum number of rows to set  
***MAX\_N*** The maximum number of columns to set  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 1084 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `MATRIX_DUPLICATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.34.2.18 subroutine MATRIX\_VECTOR::MATRIX\_NUMBER\_NON\_ZEROS\_SET**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in)  
NUMBER_NON_ZEROS, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets/changes the number of non zeros for a matrix.

**Parameters:**

***MATRIX*** A pointer to the matrix  
***NUMBER\_NON\_ZEROS*** The number of non zeros in the matrix to set  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 1030 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `MATRIX_BLOCK_STORAGE_TYPE`,    `MATRIX_COLUMN_MAJOR_STORAGE_TYPE`,    `MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE`,    `MATRIX_COMPRESSED_ROW_STORAGE_TYPE`,    `MATRIX_DIAGONAL_STORAGE_TYPE`,    `MATRIX_ROW_COLUMN_STORAGE_TYPE`, and `MATRIX_ROW_MAJOR_STORAGE_TYPE`.

Referenced by `MATRIX_DUPLICATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.34.2.19 subroutine MATRIX\_VECTOR::MATRIX\_OUTPUT (INTEGER(INTG),intent(in) *ID*, TYPE(MATRIX\_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Sets/changes the size of a matrix.

**Parameters:**

*ID* The ID to output to

*MATRIX* A pointer to the matrix

*ERR* The error code

*ERROR* The error string

Definition at line 1144 of file matrix\_vector.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_ROUTINES::EXITS(),    MATRIX\_BLOCK\_STORAGE\_TYPE,    MATRIX\_COLUMN\_MAJOR\_STORAGE\_TYPE,    MATRIX\_COMPRESSED\_COLUMN\_STORAGE\_TYPE,    MATRIX\_COMPRESSED\_ROW\_STORAGE\_TYPE,    MATRIX\_DIAGONAL\_STORAGE\_TYPE,    MATRIX\_ROW\_COLUMN\_STORAGE\_TYPE,    MATRIX\_ROW\_MAJOR\_STORAGE\_TYPE,    MATRIX\_VECTOR\_DP\_TYPE,    MATRIX\_VECTOR\_INTG\_TYPE,    MATRIX\_VECTOR\_L\_TYPE,    MATRIX\_VECTOR\_SP\_TYPE, and INPUT\_OUTPUT::WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES.

Here is the call graph for this function:

**6.34.2.20 subroutine MATRIX\_VECTOR::MATRIX\_SIZE\_SET (TYPE(MATRIX\_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(in) *M*, INTEGER(INTG),intent(in) *N*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Sets/changes the size of a matrix.

**Parameters:**

*MATRIX* A pointer to the matrix

*M* The number of rows to set

*N* The number of columns to set

*ERR* The error code

*ERROR* The error string

Definition at line 1265 of file matrix\_vector.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_ROUTINES::EXITS().

Referenced by MATRIX\_DUPLICATE().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.34.2.21 subroutine MATRIX\_VECTOR::MATRIX\_STORAGE\_LOCATION\_FIND**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) I,  
 INTEGER(INTG),intent(in) J, INTEGER(INTG),intent(out) LOCATION,  
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,  
 *)`

Returns the storage location in the data array of a matrix that correponds to location I,J. If the location does not exist the routine returns zero.

**Parameters:**

**MATRIX** A pointer to the matrix

**I** The row number of the location to find

**J** The column number of the location to find

**LOCATION** On return the location of the specified row and column in the matrix data. If the row and column does not exist in the matrix then zero is returned.

**ERR** The error code

**ERROR** The error string

Definition at line 1313 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `MATRIX_BLOCK_STORAGE_TYPE`,    `MATRIX_COLUMN_MAJOR_STORAGE_TYPE`,    `MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE`,    `MATRIX_COMPRESSED_ROW_STORAGE_TYPE`,    `MATRIX_DIAGONAL_STORAGE_TYPE`,    `MATRIX_ROW_COLUMN_STORAGE_TYPE`, and `MATRIX_ROW_MAJOR_STORAGE_TYPE`.

Referenced by `MATRIX_VALUES_ADD_DP()`, `MATRIX_VALUES_ADD_DP1()`, `MATRIX_VALUES_ADD_DP2()`, `MATRIX_VALUES_ADD_INTG()`, `MATRIX_VALUES_ADD_INTG1()`, `MATRIX_VALUES_ADD_INTG2()`, `MATRIX_VALUES_ADD_L()`, `MATRIX_VALUES_ADD_L1()`, `MATRIX_VALUES_ADD_L2()`, `MATRIX_VALUES_ADD_SP()`, `MATRIX_VALUES_ADD_SP1()`, `MATRIX_VALUES_ADD_SP2()`, `MATRIX_VALUES_GET_DP()`, `MATRIX_VALUES_GET_DP1()`, `MATRIX_VALUES_GET_DP2()`, `MATRIX_VALUES_GET_INTG()`, `MATRIX_VALUES_GET_INTG1()`, `MATRIX_VALUES_GET_INTG2()`, `MATRIX_VALUES_GET_L()`, `MATRIX_VALUES_GET_L1()`, `MATRIX_VALUES_GET_L2()`, `MATRIX_VALUES_GET_SP()`, `MATRIX_VALUES_GET_SP1()`, `MATRIX_VALUES_GET_SP2()`, `MATRIX_VALUES_SET_DP()`, `MATRIX_VALUES_SET_DP1()`, `MATRIX_VALUES_SET_DP2()`, `MATRIX_VALUES_SET_INTG()`, `MATRIX_VALUES_SET_INTG1()`, `MATRIX_VALUES_SET_INTG2()`, `MATRIX_VALUES_SET_L()`, `MATRIX_VALUES_SET_L1()`, `MATRIX_VALUES_SET_L2()`, `MATRIX_VALUES_SET_SP()`, `MATRIX_VALUES_SET_SP1()`, and `MATRIX_VALUES_SET_SP2()`.

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.34.2.22 subroutine MATRIX\_VECTOR::MATRIX\_STORAGE\_LOCATIONS\_GET**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),pointer  
 ROW_INDICES, INTEGER(INTG),dimension(:),pointer COLUMN_INDICES,  
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,  
 *)`

Gets the storage locations (sparsity pattern) of a matrix.

**Parameters:**

**MATRIX** A pointer to the matrix

**ROW\_INDICES** ROW\_INDICES(i). On return, the row index values for the matrix.

**COLUMN\_INDICES** COLUMN\_INDICES(i). On return, the column index values for the matrix.

**ERR** The error code

**ERROR** The error string

Definition at line 1433 of file matrix\_vector.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), MATRIX\_BLOCK\_STORAGE\_TYPE, MATRIX\_COLUMN\_MAJOR\_STORAGE\_TYPE, MATRIX\_COMPRESSED\_COLUMN\_STORAGE\_TYPE, MATRIX\_COMPRESSED\_ROW\_STORAGE\_TYPE, MATRIX\_DIAGONAL\_STORAGE\_TYPE, MATRIX\_ROW\_COLUMN\_STORAGE\_TYPE, and MATRIX\_ROW\_MAJOR\_STORAGE\_TYPE.

Here is the call graph for this function:

**6.34.2.23 subroutine MATRIX\_VECTOR::MATRIX\_STORAGE\_LOCATIONS\_SET**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the storage locations (sparsity pattern) in a matrix to that specified by the row and column indices.

#### Parameters:

**MATRIX** A pointer to the matrix

**ROW\_INDICES** ROW\_INDICES(i). The row index values for the sparsity pattern.

**COLUMN\_INDICES** COLUMN\_INDICES(i). The column index values for the sparsity pattern.

**ERR** The error code

**ERROR** The error string

Definition at line 1489 of file matrix\_vector.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), MATRIX\_BLOCK\_STORAGE\_TYPE, MATRIX\_COLUMN\_MAJOR\_STORAGE\_TYPE, MATRIX\_COMPRESSED\_COLUMN\_STORAGE\_TYPE, MATRIX\_COMPRESSED\_ROW\_STORAGE\_TYPE, MATRIX\_DIAGONAL\_STORAGE\_TYPE, MATRIX\_ROW\_COLUMN\_STORAGE\_TYPE, and MATRIX\_ROW\_MAJOR\_STORAGE\_TYPE.

Referenced by MATRIX\_DUPLICATE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.34.2.24 subroutine MATRIX\_VECTOR::MATRIX\_STORAGE\_TYPE\_GET**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(out) STORAGE_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets the storage type for a matrix.

#### Parameters:

**MATRIX** A pointer to the matrix

**STORAGE\_TYPE** On return, the storage type of the matrix.

See also:

[MATRIX\\_VECTOR::StorageTypes](#), [MATRIX\\_VECTOR](#)

**ERR** The error code

**ERROR** The error string

Definition at line 1709 of file matrix\_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.34.2.25 subroutine MATRIX\_VECTOR::MATRIX\_STORAGE\_TYPE\_SET (TYPE(MATRIX\_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) STORAGE\_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets/changes the storage type for a matrix.

**Parameters:**

**MATRIX** A pointer to the matrix

**STORAGE\_TYPE** The storage type to set.

See also:

[MATRIX\\_VECTOR::StorageTypes](#), [MATRIX\\_VECTOR](#)

**ERR** The error code

**ERROR** The error string

Definition at line 1742 of file matrix\_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_BLOCK_STORAGE_TYPE`, `MATRIX_COLUMN_MAJOR_STORAGE_TYPE`, `MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE`, `MATRIX_COMPRESSED_ROW_STORAGE_TYPE`, `MATRIX_DIAGONAL_STORAGE_TYPE`, `MATRIX_ROW_COLUMN_STORAGE_TYPE`, and `MATRIX_ROW_MAJOR_STORAGE_TYPE`.

Referenced by `MATRIX_DUPLICATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.34.2.26 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD\_DP (TYPE(MATRIX\_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:,),intent(in) ROW\_INDICES, INTEGER(INTG),dimension(:,),intent(in) COLUMN\_INDICES, REAL(DP),dimension(:,),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Adds values to a double precision real matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I,J)=\text{MATRIX}(I,J)+\text{VALUE}$ .

**Parameters:**

**MATRIX** A pointer to the matrix

**ROW\_INDICES** ROW\_INDICES(i). The row index for the i'th value to add

**COLUMN\_INDICES** COLUMN\_INDICES(i). The column index for the i'th value to add

**VALUES** VALUES(i). The value of the i'th value to add

**ERR** The error code

**ERROR** The error string

Definition at line 2166 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(),    MATRIX\_STORAGE\_LOCATION\_FIND(),   and    MATRIX\_VECTOR\_DP\_-TYPE.

Here is the call graph for this function:

**6.34.2.27 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD\_DP1**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX,  
 INTEGER(INTG),intent(in) COLUMN_INDEX, REAL(DP),intent(in) VALUE,  
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,  
 *)`

Adds a value to a double precision real matrix at the location specified by the row and column index i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.

**Parameters:**

**MATRIX** A pointer to the matrix

**ROW\_INDEX** The row index for the value to add

**COLUMN\_INDEX** The column index for the value to add

**VALUE** The value to add

**ERR** The error code

**ERROR** The error string

Definition at line 2232 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(),    MATRIX\_STORAGE\_LOCATION\_FIND(),   and    MATRIX\_VECTOR\_DP\_-TYPE.

Here is the call graph for this function:

**6.34.2.28 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD\_DP2**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in)  
 ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES,  
 REAL(DP),dimension(:, :, ),intent(in) VALUES, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING_STRING),intent(out) ERROR, *)`

Adds a matrix of values to a double precision real matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.

**Parameters:**

**MATRIX** A pointer to the matrix

**ROW\_INDICES** ROW\_INDICES(i). The row index for the ij'th value to add

**COLUMN\_INDICES** COLUMN\_INIDICES(j). The column index for the ij'th value to add

**VALUES** VALUES(i,j). The value of the ij'th value to add

**ERR** The error code

**ERROR** The error string

Definition at line 2282 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(),    MATRIX\_STORAGE\_LOCATION\_FIND(), and    MATRIX\_VECTOR\_DP\_-TYPE.

Here is the call graph for this function:

**6.34.2.29 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD\_INTG  
(TYPE(MATRIX\_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in)  
ROW\_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN\_INDICES,  
INTEGER(INTG),dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Adds values to an integer matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.

**Parameters:**

**MATRIX** A pointer to the matrix

**ROW\_INDICES** ROW\_INDICES(i). The row index for the i'th value to add

**COLUMN\_INDICES** COLUMN\_INIDICES(i). The column index for the i'th value to add

**VALUES** VALUES(i). The value of the i'th value to add

**ERR** The error code

**ERROR** The error string

Definition at line 1794 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(),    MATRIX\_STORAGE\_LOCATION\_FIND(), and    MATRIX\_VECTOR\_INTG\_-TYPE.

Here is the call graph for this function:

**6.34.2.30 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD\_INTG1  
(TYPE(MATRIX\_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW\_INDEX,  
INTEGER(INTG),intent(in) COLUMN\_INDEX, INTEGER(INTG),intent(in) VALUE,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*)**

Adds a value to an integer matrix at the location specified by the row and column index i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.

**Parameters:**

**MATRIX** A pointer to the matrix  
**ROW\_INDEX** The row index for the value to add  
**COLUMN\_INDEX** The column index for the value to add  
**VALUE** The value to add  
**ERR** The error code  
**ERROR** The error string

Definition at line 1860 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(), MATRIX\_STORAGE\_LOCATION\_FIND(), and MATRIX\_VECTOR\_INTG\_-TYPE.

Here is the call graph for this function:

**6.34.2.31 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD\_INTG2**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, INTEGER(INTG),dimension(:, :, intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Adds a matrix of values to an integer matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.

**Parameters:**

**MATRIX** A pointer to the matrix  
**ROW\_INDICES** ROW\_INDICES(i). The row index for the ij'th value to add  
**COLUMN\_INDICES** COLUMN\_INIDICES(j). The column index for the ij'th value to add  
**VALUES** VALUES(i,j). The value of the ij'th value to add  
**ERR** The error code  
**ERROR** The error string

Definition at line 1910 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(), MATRIX\_STORAGE\_LOCATION\_FIND(), and MATRIX\_VECTOR\_INTG\_-TYPE.

Here is the call graph for this function:

**6.34.2.32 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD\_L**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, LOGICAL,dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Adds values to a logical matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J).OR.VALUE.

**Parameters:**

**MATRIX** A pointer to the matrix

**ROW\_INDICES** ROW\_INDICES(i). The row index for the i'th value to add

**COLUMN\_INDICES** COLUMN\_INDICES(i). The column index for the i'th value to add

**VALUES** VALUES(i). The value of the i'th value to add

**ERR** The error code

**ERROR** The error string

Definition at line 2352 of file matrix\_vector.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_ROUTINES::EXITS(), MATRIX\_STORAGE\_LOCATION\_FIND(), and MATRIX\_VECTOR\_L\_TYPE.

Here is the call graph for this function:

**6.34.2.33 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD\_L1**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX,  
 INTEGER(INTG),intent(in) COLUMN_INDEX, LOGICAL,intent(in) VALUE,  
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,  
 *)`

Adds a value to a logical matrix at the location specified by the row and column index i.e., MATRIX(I,J)=MATRIX(I,J).OR.VALUE.

**Parameters:**

**MATRIX** A pointer to the matrix

**ROW\_INDEX** The row index for the value to add

**COLUMN\_INDEX** The column index for the value to add

**VALUE** The value to add

**ERR** The error code

**ERROR** The error string

Definition at line 2418 of file matrix\_vector.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_ROUTINES::EXITS(), MATRIX\_STORAGE\_LOCATION\_FIND(), and MATRIX\_VECTOR\_L\_TYPE.

Here is the call graph for this function:

**6.34.2.34 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD\_L2**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:,),intent(in)  
 ROW_INDICES, INTEGER(INTG),dimension(:,),intent(in) COLUMN_INDICES,  
 LOGICAL,dimension(:, :,),intent(in) VALUES, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING_STRING),intent(out) ERROR, *)`

Adds a matrix of values to a logical matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J).OR.VALUE.

**Parameters:**

**MATRIX** A pointer to the matrix

**ROW\_INDICES** ROW\_INDICES(i). The row index for the ij'th value to add

**COLUMN\_INDICES** COLUMN\_INIDICES(j). The column index for the ij'th value to add

**VALUES** VALUES(i,j). The value of the ij'th value to add

**ERR** The error code

**ERROR** The error string

Definition at line 2468 of file matrix\_vector.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), MATRIX\_STORAGE\_LOCATION\_FIND(), and MATRIX\_VECTOR\_L\_TYPE.

Here is the call graph for this function:

**6.34.2.35 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD\_SP**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in)`  
`ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES,`  
`REAL(SP),dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Adds values to a single precision real matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I,J)=\text{MATRIX}(I,J)+\text{VALUE}$ .

#### Parameters:

**MATRIX** A pointer to the matrix

**ROW\_INDICES** ROW\_INDICES(i). The row index for the i'th value to add

**COLUMN\_INDICES** COLUMN\_INIDICES(i). The column index for the i'th value to add

**VALUES** VALUES(i). The value of the i'th value to add

**ERR** The error code

**ERROR** The error string

Definition at line 1980 of file matrix\_vector.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(),    MATRIX\_STORAGE\_LOCATION\_FIND(), and    MATRIX\_VECTOR\_SP\_-TYPE.

Here is the call graph for this function:

**6.34.2.36 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD\_SP1**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX,`  
`INTEGER(INTG),intent(in) COLUMN_INDEX, REAL(SP),intent(in) VALUE,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`  
`*)`

Adds a value to a single precision real matrix at the location specified by the row and column index i.e.,  $\text{MATRIX}(I,J)=\text{MATRIX}(I,J)+\text{VALUE}$ .

#### Parameters:

**MATRIX** A pointer to the matrix

**ROW\_INDEX** The row index for the value to add

**COLUMN\_INDEX** The column index for the value to add

**VALUE** The value to add

**ERR** The error code

**ERROR** The error string

Definition at line 2046 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `MATRIX_STORAGE_LOCATION_FIND()`,    and    `MATRIX_VECTOR_SP_TYPE`.

Here is the call graph for this function:

```
6.34.2.37 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD_SP2
 (TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in)
 ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES,
 REAL(SP),dimension(:, :),intent(in) VALUES, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *)
```

Adds a matrix of values to a single precision real matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I,J)=\text{MATRIX}(I,J)+\text{VALUE}$ .

#### Parameters:

**MATRIX** A pointer to the matrix

**ROW\_INDICES** `ROW_INDICES(i)`. The row index for the  $ij$ 'th value to add

**COLUMN\_INDICES** `COLUMN_INIDICES(j)`. The column index for the  $ij$ 'th value to add

**VALUES** `VALUES(i,j)`. The value of the  $ij$ 'th value to add

**ERR** The error code

**ERROR** The error string

Definition at line 2096 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `MATRIX_STORAGE_LOCATION_FIND()`,    and    `MATRIX_VECTOR_SP_TYPE`.

Here is the call graph for this function:

```
6.34.2.38 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET_DP
 (TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in)
 ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES,
 REAL(DP),dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *)
```

Gets values in a double precision real matrix at the location specified by the row and column indices i.e.,  $\text{VALUE}=\text{MATRIX}(I,J)$ .

#### Parameters:

**MATRIX** A pointer to the matrix

**ROW\_INDICES** `ROW_INDICES(i)`. The row index for the  $i$ 'th value to get

**COLUMN\_INDICES** COLUMN\_INIDICES(i). The column index for the i'th value to get  
**VALUES** VALUES(i). On return the value of the i'th value to get  
**ERR** The error code  
**ERROR** The error string

Definition at line 2898 of file matrix\_vector.f90.

References KINDS::\_DP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), MATRIX\_STORAGE\_LOCATION\_FIND(), and MATRIX\_VECTOR\_DP\_TYPE.

Here is the call graph for this function:

**6.34.2.39 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_GET\_DP1**  
 (TYPE(MATRIX\_TYPE),pointer **MATRIX**, INTEGER(INTG),intent(in) **ROW\_INDEX**,  
 INTEGER(INTG),intent(in) **COLUMN\_INDEX**, REAL(DP),intent(out) **VALUE**,  
 INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**,  
 \*)

Gets a value in a double precision real matrix at the location specified by the row and column index i.e., VALUE=MATRIX(I,J).

#### Parameters:

**MATRIX** A pointer to the matrix  
**ROW\_INDEX** The row index of the value to get  
**COLUMN\_INDEX** The column index of the value to get  
**VALUE** On return the value in the matrix at the specified row and column  
**ERR** The error code  
**ERROR** The error string

Definition at line 2962 of file matrix\_vector.f90.

References KINDS::\_DP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), MATRIX\_STORAGE\_LOCATION\_FIND(), and MATRIX\_VECTOR\_DP\_TYPE.

Here is the call graph for this function:

**6.34.2.40 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_GET\_DP2**  
 (TYPE(MATRIX\_TYPE),pointer **MATRIX**, INTEGER(INTG),dimension(:),intent(in)  
**ROW\_INDICES**, INTEGER(INTG),dimension(:),intent(in) **COLUMN\_INDICES**,  
 REAL(DP),dimension(:, :, ),intent(out) **VALUES**, INTEGER(INTG),intent(out) **ERR**,  
 TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

Gets a matrix of values in a double precision real matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).

#### Parameters:

**MATRIX** A pointer to the matrix  
**ROW\_INDICES** ROW\_INDICES(i). The row index for the ij'th value to get

**COLUMN\_INDICES** COLUMN\_INIDICES(j). The column index for the ij'th value to get  
**VALUES** VALUES(i,j). On return the value of the ij'th value to get  
**ERR** The error code  
**ERROR** The error string

Definition at line 3010 of file matrix\_vector.f90.

References KINDS::DP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), MATRIX\_STORAGE\_LOCATION\_FIND(), and MATRIX\_VECTOR\_DP\_TYPE.

Here is the call graph for this function:

```
6.34.2.41 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET_INTG
 (TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in)
 ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES,
 INTEGER(INTG),dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out)
 ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)
```

Gets the values in an integer matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).

#### Parameters:

**MATRIX** A pointer to the matrix  
**ROW\_INDICES** ROW\_INDICES(i). The row index for the i'th value to get  
**COLUMN\_INDICES** COLUMN\_INIDICES(i). The column index for the i'th value to get  
**VALUES** VALUES(i). On return the value of the i'th value to get  
**ERR** The error code  
**ERROR** The error string

Definition at line 2538 of file matrix\_vector.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), MATRIX\_STORAGE\_LOCATION\_FIND(), and MATRIX\_VECTOR\_INTG\_TYPE.

Here is the call graph for this function:

```
6.34.2.42 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET_INTG1
 (TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX,
 INTEGER(INTG),intent(in) COLUMN_INDEX, INTEGER(INTG),intent(out) VALUE,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *)
```

Gets a value in an integer matrix at the location specified by the row and column index i.e., VALUE=MATRIX(I,J).

#### Parameters:

**MATRIX** A pointer to the matrix  
**ROW\_INDEX** The row index of the value to get

**COLUMN\_INDEX** The column index of the value to get  
**VALUE** On return the value in the matrix at the specified row and column  
**ERR** The error code  
**ERROR** The error string

Definition at line 2602 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_INTG_TYPE()`.

Here is the call graph for this function:

**6.34.2.43 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_GET\_INTG2**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, INTEGER(INTG),dimension(:, :, ),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets the matrix of values in an integer matrix at the location specified by the row and column indices i.e., `VALUE=MATRIX(I,J)`.

#### Parameters:

**MATRIX** A pointer to the matrix  
**ROW\_INDICES** `ROW_INDICES(i)`. The row index for the  $i^{\text{th}}$  value to get  
**COLUMN\_INDICES** `COLUMN_INIDICES(j)`. The column index for the  $i^{\text{th}}$  value to get  
**VALUES** `VALUES(i,j)`. On return the value of the  $i^{\text{th}}$  value to get  
**ERR** The error code  
**ERROR** The error string

Definition at line 2650 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_INTG_TYPE()`.

Here is the call graph for this function:

**6.34.2.44 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_GET\_L**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, LOGICAL,dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets values in a logical matrix at the location specified by the row and column indices i.e., `VALUE=MATRIX(I,J)`.

#### Parameters:

**MATRIX** A pointer to the matrix  
**ROW\_INDICES** `ROW_INDICES(i)`. The row index for the  $i^{\text{th}}$  value to get

**COLUMN\_INDICES** COLUMN\_INIDICES(i). The column index for the i'th value to get

**VALUES** VALUES(i). On return the value of the i'th value to get

**ERR** The error code

**ERROR** The error string

Definition at line 3078 of file matrix\_vector.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_ROUTINES::EXITS(), MATRIX\_STORAGE\_LOCATION\_FIND(), and MATRIX\_VECTOR\_L\_TYPE.

Here is the call graph for this function:

**6.34.2.45 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_GET\_L1**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX,  
 INTEGER(INTG),intent(in) COLUMN_INDEX, LOGICAL,intent(out) VALUE,  
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,  
 *)`

Gets a value in a logical matrix at the location specified by the row and column index i.e.,  
`VALUE=MATRIX(I,J).`

#### Parameters:

**MATRIX** A pointer to the matrix

**ROW\_INDEX** The row index of the value to get

**COLUMN\_INDEX** The column index of the value to get

**VALUE** On return the value in the matrix at the specified row and column

**ERR** The error code

**ERROR** The error string

Definition at line 3142 of file matrix\_vector.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_ROUTINES::EXITS(), MATRIX\_STORAGE\_LOCATION\_FIND(), and MATRIX\_VECTOR\_L\_TYPE.

Here is the call graph for this function:

**6.34.2.46 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_GET\_L2**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:,),intent(in)  
 ROW_INDICES, INTEGER(INTG),dimension(:,),intent(in) COLUMN_INDICES,  
 LOGICAL,dimension(:, :,),intent(out) VALUES, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets a matrix of values in a logical matrix at the location specified by the row and column indices i.e.,  
`VALUE=MATRIX(I,J).`

#### Parameters:

**MATRIX** A pointer to the matrix

**ROW\_INDICES** ROW\_INDICES(i). The row index for the ij'th value to get

**COLUMN\_INDICES** COLUMN\_INIDICES(j). The column index for the ij'th value to get

**VALUES** VALUES(i,j). On return the value of the ij'th value to get

**ERR** The error code

**ERROR** The error string

Definition at line 3190 of file matrix\_vector.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), MATRIX\_STORAGE\_LOCATION\_FIND(), and MATRIX\_VECTOR\_L\_TYPE.

Here is the call graph for this function:

**6.34.2.47 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_GET\_SP**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, REAL(SP),dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets values in a single precision real matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).

#### Parameters:

**MATRIX** A pointer to the matrix

**ROW\_INDICES** ROW\_INDICES(i). The row index for the i'th value to get

**COLUMN\_INDICES** COLUMN\_INDICES(i). The column index for the i'th value to get

**VALUES** VALUES(i). On return the value of the i'th value to get

**ERR** The error code

**ERROR** The error string

Definition at line 2718 of file matrix\_vector.f90.

References KINDS::SP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), MATRIX\_STORAGE\_LOCATION\_FIND(), and MATRIX\_VECTOR\_SP\_TYPE.

Here is the call graph for this function:

**6.34.2.48 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_GET\_SP1**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX, INTEGER(INTG),intent(in) COLUMN_INDEX, REAL(SP),intent(out) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets a value in a single precision real matrix at the location specified by the row and column index i.e., VALUE=MATRIX(I,J).

#### Parameters:

**MATRIX** A pointer to the matrix

**ROW\_INDEX** The row index of the value to get

**COLUMN\_INDEX** The column index of the value to get

**VALUE** On return the value in the matrix at the specified row and column

**ERR** The error code

**ERROR** The error string

Definition at line 2782 of file matrix\_vector.f90.

References `KINDS::_SP`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_SP_TYPE`.

Here is the call graph for this function:

**6.34.2.49 subroutine `MATRIX_VECTOR::MATRIX_VALUES_GET_SP2`**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, REAL(SP),dimension(:, :, ),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets a matrix of values in a single precision real matrix at the location specified by the row and column indices i.e.,  $\text{VALUE} = \text{MATRIX}(I, J)$ .

#### Parameters:

***MATRIX*** A pointer to the matrix

***ROW\_INDICES*** `ROW_INDICES(i)`. The row index for the  $i^{\text{th}}$  value to get

***COLUMN\_INDICES*** `COLUMN_INDICES(j)`. The column index for the  $i^{\text{th}}$  value to get

***VALUES*** `VALUES(i, j)`. On return the value of the  $i^{\text{th}}$  value to get

**ERR** The error code

**ERROR** The error string

Definition at line 2830 of file matrix\_vector.f90.

References `KINDS::_SP`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_SP_TYPE`.

Here is the call graph for this function:

**6.34.2.50 subroutine `MATRIX_VECTOR::MATRIX_VALUES_SET_DP`**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, REAL(DP),dimension(:, :, ),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the values in a double precision real matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I, J) = \text{VALUE}$ .

#### Parameters:

***MATRIX*** A pointer to the matrix to set.

***ROW\_INDICES*** `ROW_INDICES(i)`. The row index of the  $i^{\text{th}}$  value to set

***COLUMN\_INDICES*** `COLUMN_INDICES(i)`. The column index of the  $i^{\text{th}}$  value to set

***VALUES*** `VALUES(i)`. The value of the  $i^{\text{th}}$  value to set

**ERR** The error code

**ERROR** The error string

Definition at line 3630 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_DP_TYPE`.

Here is the call graph for this function:

**6.34.2.51 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_SET\_DP1**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX,`  
`INTEGER(INTG),intent(in) COLUMN_INDEX, REAL(DP),intent(in) VALUE,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`  
`*)`

Sets a value in a double precision real matrix at the location specified by the row and column index i.e., `MATRIX(I,J)=VALUE`.

#### Parameters:

**MATRIX** A pointer to the matrix

**ROW\_INDEX** The row index of the value to set

**COLUMN\_INDEX** The column index of the value to set

**VALUE** The value to set

**ERR** The error code

**ERROR** The error string

Definition at line 3696 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_DP_TYPE`.

Here is the call graph for this function:

**6.34.2.52 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_SET\_DP2**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:,),intent(in) ROW_INDICES,`  
`INTEGER(INTG),dimension(:,),intent(in) COLUMN_INDICES,`  
`REAL(DP),dimension(:, :,),intent(in) VALUES, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the matrix of values in a double precision real matrix at the location specified by the row and column indices i.e., `MATRIX(I,J)=VALUE`.

#### Parameters:

**MATRIX** A pointer to the matrix to set.

**ROW\_INDICES** `ROW_INDICES(i)`. The row index of the ij'th value to set

**COLUMN\_INDICES** `COLUMN_INDICES(j)`. The column index of the ij'th value to set

**VALUES** `VALUES(i,j)`. The value of the ij'th value to set

**ERR** The error code

**ERROR** The error string

Definition at line 3746 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_DP_TYPE`.

Here is the call graph for this function:

**6.34.2.53 subroutine `MATRIX_VECTOR::MATRIX_VALUES_SET_INTG`**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, INTEGER(INTG),dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the values in an integer matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I,J)=\text{VALUE}$ .

#### Parameters:

***MATRIX*** A pointer to the matrix

***ROW\_INDICES*** *ROW\_INDICES*(i). The row index for the i'th value to set

***COLUMN\_INDICES*** *COLUMN\_INDICES*(i). The column index for the i'th value to set

***VALUES*** *VALUES*(i). The value of the i'th value to set

**ERR** The error code

**ERROR** The error string

Definition at line 3258 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_INTG_TYPE`.

Here is the call graph for this function:

**6.34.2.54 subroutine `MATRIX_VECTOR::MATRIX_VALUES_SET_INTG1`**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX, INTEGER(INTG),intent(in) COLUMN_INDEX, INTEGER(INTG),intent(in) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets a value in an integer matrix at the location specified by the row and column index i.e.,  $\text{MATRIX}(I,J)=\text{VALUE}$ .

#### Parameters:

***MATRIX*** A pointer to the matrix

***ROW\_INDEX*** The row index of the value to set

***COLUMN\_INDEX*** The column index of the value to set

***VALUE*** The value to set

**ERR** The error code

**ERROR** The error string

Definition at line 3324 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_INTG_TYPE()`.

Here is the call graph for this function:

**6.34.2.55 subroutine `MATRIX_VECTOR::MATRIX_VALUES_SET_INTG2`**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, INTEGER(INTG),dimension(:, :, :),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the matrix of values in an integer matrix at the location specified by the row and column indices i.e., `MATRIX(I,J)=VALUE`.

#### Parameters:

**MATRIX** A pointer to the matrix

**ROW\_INDICES** `ROW_INDICES(i)`. The row index for the ij'th value to set

**COLUMN\_INDICES** `COLUMN_INDICES(j)`. The column index for the ij'th value to set

**VALUES** `VALUES(i,j)`. The value of the ij'th value to set

**ERR** The error code

**ERROR** The error string

Definition at line 3374 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_INTG_TYPE()`.

Here is the call graph for this function:

**6.34.2.56 subroutine `MATRIX_VECTOR::MATRIX_VALUES_SET_L`**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, LOGICAL,dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the values in a logical matrix at the location specified by the row and column indices i.e., `MATRIX(I,J)=VALUE`.

#### Parameters:

**MATRIX** A pointer to the matrix to set

**ROW\_INDICES** `ROW_INDICES(i)`. The row index of the i'th value to set

**COLUMN\_INDICES** `COLUMN_INDICES(i)`. The column index of the i'th value to set

**VALUES** `VALUES(i)`. The value of the i'th value to set

**ERR** The error code

**ERROR** The error string

Definition at line 3816 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_L_TYPE`.

Here is the call graph for this function:

**6.34.2.57 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_SET\_L1**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX,`  
`INTEGER(INTG),intent(in) COLUMN_INDEX, LOGICAL,intent(in) VALUE,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`  
`*)`

Sets a value in a logical matrix at the location specified by the row and column index i.e., `MATRIX(I,J)=VALUE`.

#### Parameters:

**MATRIX** A pointer to the matrix

**ROW\_INDEX** The row index of the value to set

**COLUMN\_INDEX** The column index of the value to set

**VALUE** The value to set

**ERR** The error code

**ERROR** The error string

Definition at line 3882 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_L_TYPE`.

Here is the call graph for this function:

**6.34.2.58 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_SET\_L2**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:,),intent(in)`  
`ROW_INDICES, INTEGER(INTG),dimension(:,),intent(in) COLUMN_INDICES,`  
`LOGICAL,dimension(:, :,),intent(in) VALUES, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the matrix of values in a logical matrix at the location specified by the row and column indices i.e., `MATRIX(I,J)=VALUE`.

#### Parameters:

**MATRIX** A pointer to the matrix to set

**ROW\_INDICES** `ROW_INDICES(i)`. The row index of the ij'th value to set

**COLUMN\_INDICES** `COLUMN_INDICES(j)`. The column index of the ij'th value to set

**VALUES** `VALUES(i,j)`. The value of the ij'th value to set

**ERR** The error code

**ERROR** The error string

Definition at line 3932 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_L_TYPE`.

Here is the call graph for this function:

**6.34.2.59 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_SET\_SP**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, REAL(SP),dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the values in a single precision real matrix at the location specified by the row and column indices i.e., `MATRIX(I,J)=VALUE`.

#### Parameters:

**MATRIX** A pointer to the matrix to set

**ROW\_INDICES** `ROW_INDICES(i)`. The row index of the i'th value to set

**COLUMN\_INDICES** `COLUMN_INDICES(i)`. The column index of the i'th value to set

**VALUES** `VALUES(i)`. The value of the i'th value to set

**ERR** The error code

**ERROR** The error string

Definition at line 3444 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_SP_TYPE`.

Here is the call graph for this function:

**6.34.2.60 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_SET\_SP1**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX, INTEGER(INTG),intent(in) COLUMN_INDEX, REAL(SP),intent(in) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the value in a single precision real matrix at the location specified by the row and column index i.e., `MATRIX(I,J)=VALUE`.

#### Parameters:

**MATRIX** A pointer to the matrix

**ROW\_INDEX** The row index of the value to set

**COLUMN\_INDEX** The column index of the value to set

**VALUE** The value to set

**ERR** The error code

**ERROR** The error string

Definition at line 3510 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_SP_TYPE`.

Here is the call graph for this function:

**6.34.2.61 subroutine `MATRIX_VECTOR::MATRIX_VALUES_SET_SP2`**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, REAL(SP),dimension(:, :, ),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the matrix of values in a single precision real matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I,J)=\text{VALUE}$ .

#### Parameters:

***MATRIX*** A pointer to the matrix to set

***ROW\_INDICES*** *ROW\_INDICES*(i). The row index of the ij'th value to set

***COLUMN\_INDICES*** *COLUMN\_INDICES*(j). The column index of the ij'th value to set

***VALUES*** *VALUES*(i,j). The value of the ij'th value to set

***ERR*** The error code

***ERROR*** The error string

Definition at line 3560 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_SP_TYPE`.

Here is the call graph for this function:

**6.34.2.62 subroutine `MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_DP`**  
`(TYPE(VECTOR_TYPE),pointer VECTOR, REAL(DP),intent(in) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets all values in a double precision vector to the specified value.

#### Parameters:

***VECTOR*** A pointer to the vector to set

***VALUE*** The value to set the vector to

***ERR*** The error code

***ERROR*** The error string

Definition at line 4082 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `MATRIX_VECTOR_DP_TYPE`.

Here is the call graph for this function:

---

**6.34.2.63 subroutine MATRIX\_VECTOR::VECTOR\_ALL\_VALUES\_SET\_INTG**  
 (*TYPE(VECTOR\_TYPE)*,pointer *VECTOR*, INTEGER(INTG),intent(in) *VALUE*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*)

Sets all values in an integer vector to the specified value.

**Parameters:**

***VECTOR*** A pointer to the vector to set  
***VALUE*** The value to set the vector to  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 4002 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(), and MATRIX\_VECTOR\_INTG\_TYPE.

Here is the call graph for this function:

**6.34.2.64 subroutine MATRIX\_VECTOR::VECTOR\_ALL\_VALUES\_SET\_L**  
 (*TYPE(VECTOR\_TYPE)*,pointer *VECTOR*, LOGICAL,intent(in) *VALUE*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*)

Sets all values in a logical vector to the specified value.

**Parameters:**

***VECTOR*** A pointer to the vector to set  
***VALUE*** The value to set the vector to  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 4122 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(), and MATRIX\_VECTOR\_L\_TYPE.

Here is the call graph for this function:

**6.34.2.65 subroutine MATRIX\_VECTOR::VECTOR\_ALL\_VALUES\_SET\_SP**  
 (*TYPE(VECTOR\_TYPE)*,pointer *VECTOR*, REAL(SP),intent(in) *VALUE*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*)

Sets all values in a single precision vector to the specified value.

**Parameters:**

***VECTOR*** A pointer to the vector to set  
***VALUE*** The value to set the vector to

**ERR** The error code

**ERROR** The error string

Definition at line 4042 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `MATRIX_VECTOR_SP_TYPE`.

Here is the call graph for this function:

**6.34.2.66 subroutine MATRIX\_VECTOR::VECTOR\_CREATE\_FINISH**  
`(TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Finishes the creation of a vector.

#### Parameters:

**VECTOR** A pointer to the vector

**ERR** The error code

**ERROR** The error string

Definition at line 4162 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `MATRIX_VECTOR_DP_TYPE`, `MATRIX_VECTOR_ID`, `MATRIX_VECTOR_INTG_TYPE`, `MATRIX_VECTOR_L_TYPE`, and `MATRIX_VECTOR_SP_TYPE`.

Referenced by `VECTOR_DUPLICATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.34.2.67 subroutine MATRIX\_VECTOR::VECTOR\_CREATE\_START**  
`(TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Starts the creation a vector.

#### Parameters:

**VECTOR** A pointer to the vector

**ERR** The error code

**ERROR** The error string

Definition at line 4216 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `MATRIX_VECTOR_DP_TYPE`, `VECTOR_FINALISE()`, and `VECTOR_INITIALISE()`.

Referenced by `VECTOR_DUPLICATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.34.2.68 subroutine MATRIX\_VECTOR::VECTOR\_DATA\_GET\_DP**  
`(TYPE(VECTOR_TYPE),pointer VECTOR, REAL(DP),dimension(:),pointer DATA,  
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,  
 *)`

Returns a pointer to the data of a double precision vector. Note: the values can be used for read operations but a [VECTOR\\_VALUES\\_SET](#) call must be used to change any values. The pointer should not be deallocated.

**Parameters:**

**VECTOR** A pointer to the vector  
**DATA** On return a pointer to the vector data  
**ERR** The error code  
**ERROR** The error string

Definition at line 4339 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(), and MATRIX\_VECTOR\_DP\_TYPE.

Here is the call graph for this function:

**6.34.2.69 subroutine MATRIX\_VECTOR::VECTOR\_DATA\_GET\_INTG**  
`(TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),pointer  
 DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)  
 ERROR, *)`

Returns a pointer to the data of an integer vector. Note: the values can be used for read operations but a [VECTOR\\_VALUES\\_SET](#) call must be used to change any values. The pointer should not be deallocated.

**Parameters:**

**VECTOR** A pointer to the vector  
**DATA** On return a pointer to the vector data  
**ERR** The error code  
**ERROR** The error string

Definition at line 4249 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(), and MATRIX\_VECTOR\_INTG\_TYPE.

Here is the call graph for this function:

**6.34.2.70 subroutine MATRIX\_VECTOR::VECTOR\_DATA\_GET\_L** (`TYPE(VECTOR_-TYPE),pointer VECTOR, LOGICAL,dimension(:),pointer DATA,  
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,  
 *)`

Returns a pointer to the data of a logical vector. Note: the values can be used for read operations but a [VECTOR\\_VALUES\\_SET](#) call must be used to change any values. The pointer should not be deallocated.

**Parameters:**

**VECTOR** A pointer to the vector  
**DATA** On return a pointer to the vector data  
**ERR** The error code  
**ERROR** The error string

Definition at line 4384 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `MATRIX_VECTOR_L_TYPE`.

Here is the call graph for this function:

**6.34.2.71 subroutine MATRIX\_VECTOR::VECTOR\_DATA\_GET\_SP (TYPE(VECTOR\_TYPE),pointer VECTOR, REAL(SP),dimension(:),pointer DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Returns a pointer to the data of a single precision vector. Note: the values can be used for read operations but a `VECTOR_VALUES_SET` call must be used to change any values. The pointer should not be deallocated.

**Parameters:**

**VECTOR** A pointer to the vector  
**DATA** On return a pointer to the vector data  
**ERR** The error code  
**ERROR** The error string

Definition at line 4294 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `MATRIX_VECTOR_SP_TYPE`.

Here is the call graph for this function:

**6.34.2.72 subroutine MATRIX\_VECTOR::VECTOR\_DATA\_TYPE\_SET (TYPE(VECTOR\_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) DATA\_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets/changes the data type of a vector.

**Parameters:**

**VECTOR** A pointer to the vector.  
**DATA\_TYPE** The data type to set.

**See also:**

[MATRIX\\_VECTOR::DataTypes](#), [MATRIX\\_VECTOR](#)

**ERR** The error code

**ERROR** The error string

Definition at line 4429 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `MATRIX_VECTOR_DP_TYPE`, `MATRIX_VECTOR_INTG_TYPE`, `MATRIX_VECTOR_L_TYPE`, and `MATRIX_VECTOR_SP_TYPE`.

Referenced by `VECTOR_DUPLICATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### **6.34.2.73 subroutine MATRIX\_VECTOR::VECTOR\_DESTROY (TYPE(VECTOR\_TYPE),pointer VECTOR, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Destroys a vector.

##### **Parameters:**

**VECTOR** A pointer to the vector

**ERR** The error code

**ERROR** The error string

Definition at line 4475 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `VECTOR_FINALISE()`.

Here is the call graph for this function:

#### **6.34.2.74 subroutine MATRIX\_VECTOR::VECTOR\_DUPLICATE (TYPE(VECTOR\_TYPE),pointer VECTOR, TYPE(VECTOR\_TYPE),pointer NEW\_VECTOR, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Duplicates a vector structure and returns a pointer to the new vector in `NEW_VECTOR`.

##### **Parameters:**

**VECTOR** A pointer to the vector to duplicate

**NEW\_VECTOR** On return a pointer to the new duplicated vector

**ERR** The error code

**ERROR** The error string

Definition at line 4503 of file matrix\_vector.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `VECTOR_CREATE_FINISH()`, `VECTOR_CREATE_START()`, `VECTOR_DATA_TYPE_SET()`, `VECTOR_FINALISE()`, and `VECTOR_SIZE_SET()`.

Here is the call graph for this function:

---

**6.34.2.75 subroutine MATRIX\_VECTOR::VECTOR\_FINALISE (TYPE(VECTOR\_-  
TYPE),pointer VECTOR, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Finalises a vector and deallocates all memory.

**Parameters:**

**VECTOR** A pointer to the vector to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 4540 of file matrix\_vector.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Referenced by VECTOR\_CREATE\_START(), VECTOR\_DESTROY(), and VECTOR\_DUPLICATE().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.34.2.76 subroutine MATRIX\_VECTOR::VECTOR\_INITIALISE (TYPE(VECTOR\_-  
TYPE),pointer VECTOR, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Initialises a vector.

**Parameters:**

**VECTOR** A pointer to the vector

**ERR** The error code

**ERROR** The error string

Definition at line 4570 of file matrix\_vector.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Referenced by VECTOR\_CREATE\_START().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.34.2.77 subroutine MATRIX\_VECTOR::VECTOR\_SIZE\_SET (TYPE(VECTOR\_-  
TYPE),pointer VECTOR, INTEGER(INTG),intent(in) N, INTEGER(INTG),intent(out)  
ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets/changes the size of a vector.

**Parameters:**

**VECTOR** A pointer to the vector

**N** The size of the vector to set

**ERR** The error code

**ERROR** The error string

Definition at line 4603 of file matrix\_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `VECTOR_DUPLICATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.34.2.78 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_GET\_DP**  
`(TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDICES, REAL(DP),dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets the values in a double precision real vector at the indices specified.

#### Parameters:

**VECTOR** A pointer to the vector

**INDICES** `INDICES(i)`. The i'th index to get

**VALUES** `VALUES(i)`. On return the i'th value to get

**ERR** The error code

**ERROR** The error string

Definition at line 4853 of file matrix\_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `MATRIX_VECTOR_DP_TYPE`.

Here is the call graph for this function:

**6.34.2.79 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_GET\_DP1**  
`(TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, REAL(DP),intent(out) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets a value in a double precision vector at the location specified by the index.

#### Parameters:

**VECTOR** A pointer to the vector

**INDEX** The index of the vector to get

**VALUE** On return the value of the vector at the specified index

**ERR** The error code

**ERROR** The error string

Definition at line 4911 of file matrix\_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `MATRIX_VECTOR_DP_TYPE`.

Here is the call graph for this function:

---

**6.34.2.80 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_GET\_INTG (TYPE(VECTOR\_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, INTEGER(INTG),dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Gets the values in an integer vector at the indices specified.

**Parameters:**

**VECTOR** A pointer to the vector  
**INDICES** INDICES(i). The i'th index to get  
**VALUES** VALUES(i). On return the i'th value to get  
**ERR** The error code  
**ERROR** The error string

Definition at line 4643 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(), and MATRIX\_VECTOR\_INTG\_TYPE.

Here is the call graph for this function:

**6.34.2.81 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_GET\_INTG1 (TYPE(VECTOR\_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, INTEGER(INTG),intent(out) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Gets a value in an integer vector at the location specified by the index.

**Parameters:**

**VECTOR** A pointer to the vector  
**INDEX** The index of the vector to get  
**VALUE** On return the value of the vector at the specified index  
**ERR** The error code  
**ERROR** The error string

Definition at line 4701 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(), and MATRIX\_VECTOR\_INTG\_TYPE.

Here is the call graph for this function:

**6.34.2.82 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_GET\_L (TYPE(VECTOR\_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, LOGICAL,dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Gets the values in a logical real vector at the indices specified.

**Parameters:**

**VECTOR** A pointer to the vector

**INDICES** INDICES(i). The i'th index to get  
**VALUES** VALUES(i). On return the i'th value to get  
**ERR** The error code  
**ERROR** The error string

Definition at line 4958 of file matrix\_vector.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and MATRIX\_VECTOR\_L\_TYPE.

Here is the call graph for this function:

**6.34.2.83 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_GET\_L1**  
 (TYPE(VECTOR\_TYPE),pointer **VECTOR**, INTEGER(INTG),intent(in)  
**INDEX**, LOGICAL,intent(out) **VALUE**, INTEGER(INTG),intent(out) **ERR**,  
 TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

Gets a value in a logical vector at the location specified by the index.

**Parameters:**

**VECTOR** A pointer to the vector  
**INDEX** The index of the vector to get  
**VALUE** On return the value of the vector at the specified index  
**ERR** The error code  
**ERROR** The error string

Definition at line 5016 of file matrix\_vector.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and MATRIX\_VECTOR\_L\_TYPE.

Here is the call graph for this function:

**6.34.2.84 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_GET\_SP**  
 (TYPE(VECTOR\_TYPE),pointer **VECTOR**, INTEGER(INTG),dimension(:),intent(in)  
**INDICES**, REAL(SP),dimension(:),intent(out) **VALUES**, INTEGER(INTG),intent(out)  
**ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

Gets the values in a single precision real vector at the indices specified.

**Parameters:**

**VECTOR** A pointer to the vector  
**INDICES** INDICES(i). The i'th index to get  
**VALUES** VALUES(i). On return the i'th value to get  
**ERR** The error code  
**ERROR** The error string

Definition at line 4748 of file matrix\_vector.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and MATRIX\_VECTOR\_SP\_TYPE.

Here is the call graph for this function:

---

**6.34.2.85 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_GET\_SP1**  
`(TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, REAL(SP),intent(out) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets a value in a single precision vector at the location specified by the index.

**Parameters:**

**VECTOR** A pointer to the vector  
**INDEX** The index of the vector to get  
**VALUE** On return the value of the vector at the specified index  
**ERR** The error code  
**ERROR** The error string

Definition at line 4806 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(), and MATRIX\_VECTOR\_SP\_TYPE.

Here is the call graph for this function:

**6.34.2.86 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_SET\_DP**  
`(TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, REAL(DP),dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the values in a double precision vector at the specified indices.

**Parameters:**

**VECTOR** A pointer to the vector  
**INDICES** INDICES(i). The i'th index to set  
**VALUES** VALUES(i). The i'th value to set  
**ERR** The error code  
**ERROR** The error string

Definition at line 5273 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(), and MATRIX\_VECTOR\_DP\_TYPE.

Here is the call graph for this function:

**6.34.2.87 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_SET\_DP1**  
`(TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, REAL(DP),intent(in) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets a value in a double precision vector at the specified index.

**Parameters:**

**VECTOR** A pointer to the vector

**INDEX** The index to set

**VALUE** The value to set at the specified index

**ERR** The error code

**ERROR** The error string

Definition at line 5331 of file matrix\_vector.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), and MATRIX\_VECTOR\_DP\_TYPE.

Here is the call graph for this function:

**6.34.2.88 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_SET\_INTG (TYPE(VECTOR\_-TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, INTEGER(INTG),dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets the values in an integer vector at the specified indices.

#### Parameters:

**VECTOR** A pointer to the vector

**INDICES** INDICES(i). The i'th index to set

**VALUES** VALUES(i). The i'th value to set

**ERR** The error code

**ERROR** The error string

Definition at line 5063 of file matrix\_vector.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), and MATRIX\_VECTOR\_INTG\_TYPE.

Here is the call graph for this function:

**6.34.2.89 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_SET\_INTG1 (TYPE(VECTOR\_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, INTEGER(INTG),intent(in) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets a value in an integer vector at the specified index.

#### Parameters:

**VECTOR** A pointer to the vector

**INDEX** The index to set

**VALUE** The value to set at the specified index

**ERR** The error code

**ERROR** The error string

Definition at line 5121 of file matrix\_vector.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), and MATRIX\_VECTOR\_INTG\_TYPE.

Here is the call graph for this function:

---

**6.34.2.90 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_SET\_L (TYPE(VECTOR\_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, LOGICAL,dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets the values in a logical vector at the specified indices.

**Parameters:**

**VECTOR** A pointer to the vector  
**INDICES** INDICES(i). The i'th index to set  
**VALUES** VALUES(i). The i'th value to set  
**ERR** The error code  
**ERROR** The error string

Definition at line 5378 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(), and MATRIX\_VECTOR\_L\_TYPE.

Here is the call graph for this function:

**6.34.2.91 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_SET\_L1 (TYPE(VECTOR\_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, LOGICAL,intent(in) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets a value in a logical vector at the specified index.

**Parameters:**

**VECTOR** A pointer to the vector  
**INDEX** The index to set  
**VALUE** The value to set at the specified index  
**ERR** The error code  
**ERROR** The error string

Definition at line 5436 of file matrix\_vector.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(), and MATRIX\_VECTOR\_L\_TYPE.

Here is the call graph for this function:

**6.34.2.92 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_SET\_SP (TYPE(VECTOR\_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, REAL(SP),dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets the values in a single precision vector at the specified indices.

**Parameters:**

**VECTOR** A pointer to the vector

**INDICES** INDICES(i). The i'th index to set

**VALUES** VALUES(i). The i'th value to set

**ERR** The error code

**ERROR** The error string

Definition at line 5168 of file matrix\_vector.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and MATRIX\_VECTOR\_SP\_TYPE.

Here is the call graph for this function:

**6.34.2.93 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_SET\_SP1**  
**(TYPE(VECTOR\_TYPE),pointer VECTOR, INTEGER(INTG),intent(in)**  
**INDEX, REAL(SP),intent(in) VALUE, INTEGER(INTG),intent(out) ERR,**  
**TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets a value in a single precision vector at the specified index.

#### Parameters:

**VECTOR** A pointer to the vector

**INDEX** The index to set

**VALUE** The value to set at the specified index

**ERR** The error code

**ERROR** The error string

Definition at line 5226 of file matrix\_vector.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and MATRIX\_VECTOR\_SP\_TYPE.

Here is the call graph for this function:

### 6.34.3 Variable Documentation

**6.34.3.1 INTEGER(INTG),save MATRIX\_VECTOR::MATRIX\_VECTOR\_ID = 1**

Definition at line 174 of file matrix\_vector.f90.

Referenced by MATRIX\_CREATE\_FINISH(), and VECTOR\_CREATE\_FINISH().

## 6.35 MESH\_ROUTINES Namespace Reference

This module handles all mesh (node and element) routines.

### Classes

- interface [MESH\\_NUMBER\\_OF\\_COMPONENTS\\_SET](#)  
*Sets/changes the number of mesh components for a mesh.*
- interface [MESH\\_NUMBER\\_OF\\_ELEMENTS\\_SET](#)  
*Sets/changes the number of elements for a mesh.*

### Functions

- subroutine [DECOMPOSITION\\_CREATE\\_FINISH](#) (MESH, DECOMPOSITION, ERR, ERROR,\*)  
*Finishes the creation of a domain decomposition on a given mesh.*
- subroutine [DECOMPOSITION\\_CREATE\\_START](#) (USER\_NUMBER, MESH, DECOMPOSITION, ERR, ERROR,\*)  
*Starts the creation of a domain decomposition for a given mesh.*
- subroutine [DECOMPOSITION\\_DESTROY](#) (USER\_NUMBER, MESH, ERR, ERROR,\*)  
*Destroys a domain decomposition identified by a user number and deallocates all memory.*
- subroutine [DECOMPOSITION\\_ELEMENT\\_DOMAIN\\_CALCULATE](#) (DECOMPOSITION, ERR, ERROR,\*)  
*Calculates the element domains for a decomposition of a mesh.*
- subroutine [DECOMPOSITION\\_ELEMENT\\_DOMAIN\\_GET](#) (DECOMPOSITION, GLOBAL\_ELEMENT\_NUMBER, DOMAIN\_NUMBER, ERR, ERROR,\*)  
*Gets the domain for a given element in a decomposition of a mesh.*
- subroutine [DECOMPOSITION\\_ELEMENT\\_DOMAIN\\_SET](#) (DECOMPOSITION, GLOBAL\_ELEMENT\_NUMBER, DOMAIN\_NUMBER, ERR, ERROR,\*)  
*Sets the domain for a given element in a decomposition of a mesh.*
- subroutine [DECOMPOSITION\\_MESH\\_COMPONENT\\_NUMBER\\_GET](#) (DECOMPOSITION, MESH\_COMPONENT\_NUMBER, ERR, ERROR,\*)  
*Gets the mesh component number which will be used for the decomposition of a mesh.*
- subroutine [DECOMPOSITION\\_MESH\\_COMPONENT\\_NUMBER\\_SET](#) (DECOMPOSITION, MESH\_COMPONENT\_NUMBER, ERR, ERROR,\*)  
*Sets/changes the mesh component number which will be used for the decomposition of a mesh.*
- subroutine [DECOMPOSITION\\_NUMBER\\_OF\\_DOMAINS\\_GET](#) (DECOMPOSITION, NUMBER\_OF\_DOMAINS, ERR, ERROR,\*)  
*Gets the number of domains for a decomposition.*

- subroutine **DECOMPOSITION\_NUMBER\_OF\_DOMAINS\_SET** (DECOMPOSITION, NUMBER\_OF\_DOMAINS, ERR, ERROR,\*)
 

*Sets/changes the number of domains for a decomposition.*
- subroutine **DOMAIN\_MAPPINGS\_NODES\_FINALISE** (DOMAIN\_MAPPINGS, ERR, ERROR,\*)
 

*Finalises the node mapping in the given domain mappings.*
- subroutine **DOMAIN\_MAPPINGS\_NODES\_INITIALISE** (DOMAIN\_MAPPINGS, ERR, ERROR,\*)
 

*Initialises the node mapping in the given domain mapping.*
- subroutine **DOMAIN\_TOPOLOGY\_CALCULATE** (TOPOLOGY, ERR, ERROR,\*)
 

*Calculates the domain topology.*
- subroutine **DOMAIN\_TOPOLOGY\_INITIALISE\_FROM\_MESH** (DOMAIN, ERR, ERROR,\*)
 

*Initialises the local domain topology from the mesh topology.*
- subroutine **DOMAIN\_TOPOLOGY\_DOFS\_FINALISE** (TOPOLOGY, ERR, ERROR,\*)
 

*Finalises the dofs in the given domain topology.*
- subroutine **DOMAIN\_TOPOLOGY\_DOFS\_INITIALISE** (TOPOLOGY, ERR, ERROR,\*)
 

*Initialises the dofs data structures for a domain topology.*
- subroutine **DOMAIN\_TOPOLOGY\_ELEMENT\_FINALISE** (ELEMENT, ERR, ERROR,\*)
 

*Finalises the given domain topology element.*
- subroutine **DOMAIN\_TOPOLOGY\_ELEMENT\_INITIALISE** (ELEMENT, ERR, ERROR,\*)
 

*Initialises the given domain topology element.*
- subroutine **DOMAIN\_TOPOLOGY\_ELEMENTS\_FINALISE** (TOPOLOGY, ERR, ERROR,\*)
 

*Finalises the elements in the given domain topology.*
- subroutine **DOMAIN\_TOPOLOGY\_ELEMENTS\_INITIALISE** (TOPOLOGY, ERR, ERROR,\*)
 

*Initialises the element data structures for a domain topology.*
- subroutine **DOMAIN\_TOPOLOGY\_FINALISE** (DOMAIN, ERR, ERROR,\*)
 

*Finalises the topology in the given domain.*
- subroutine **DOMAIN\_TOPOLOGY\_INITIALISE** (DOMAIN, ERR, ERROR,\*)
 

*Initialises the topology for a given domain.*
- subroutine **DOMAIN\_TOPOLOGY\_LINE\_FINALISE** (LINE, ERR, ERROR,\*)
 

*Finalises a line in the given domain topology and deallocates all memory.*
- subroutine **DOMAIN\_TOPOLOGY\_LINE\_INITIALISE** (LINE, ERR, ERROR,\*)
 

*Initialises the line data structure for a domain topology line.*
- subroutine **DOMAIN\_TOPOLOGY\_LINES\_FINALISE** (TOPOLOGY, ERR, ERROR,\*)

*Finalises the lines in the given domain topology.*

- subroutine [DOMAIN\\_TOPOLOGY\\_LINES\\_INITIALISE](#) (TOPOLOGY, ERR, ERROR,\*)
 

*Initialises the line data structures for a domain topology.*
- subroutine [DOMAIN\\_TOPOLOGY\\_NODE\\_FINALISE](#) (NODE, ERR, ERROR,\*)
 

*Finalises the given domain topology node and deallocates all memory.*
- subroutine [DOMAIN\\_TOPOLOGY\\_NODE\\_INITIALISE](#) (NODE, ERR, ERROR,\*)
 

*Initialises the given domain topology node.*
- subroutine [DOMAIN\\_TOPOLOGY\\_NODES\\_FINALISE](#) (TOPOLOGY, ERR, ERROR,\*)
 

*Finalises the nodees in the given domain topology.*
- subroutine [DOMAIN\\_TOPOLOGY\\_NODES\\_INITIALISE](#) (TOPOLOGY, ERR, ERROR,\*)
 

*Initialises the nodes data structures for a domain topology.*
- subroutine [DOMAIN\\_TOPOLOGY\\_NODES\\_SURROUNDING\\_ELEMENTS\\_CALCULATE](#) (TOPOLOGY, ERR, ERROR,\*)
 

*Calculates the element numbers surrounding a node for a domain.*
- subroutine [MESH\\_CREATE\\_FINISH](#) (REGION, MESH, ERR, ERROR,\*)
 

*Finishes the process of creating a mesh on a region.*
- subroutine [MESH\\_CREATE\\_START](#) (USER\_NUMBER, REGION, NUMBER\_OF\_DIMENSIONS, MESH, ERR, ERROR,\*)
 

*Starts the process of creating a mesh defined by a user number with the specified NUMBER\_OF\_DIMENSIONS in the region identified by REGION.*
- subroutine [MESH\\_DESTROY](#) (USER\_NUMBER, REGION, ERR, ERROR,\*)
 

*Destroys the mesh identified by a user number on the given region and deallocates all memory.*
- subroutine [MESH\\_FINALISE](#) (MESH, ERR, ERROR,\*)
 

*Finalises a mesh and deallocates all memory.*
- subroutine [MESH\\_INITIALISE](#) (MESH, ERR, ERROR,\*)
 

*Initialises a mesh.*
- INTEGER(INTG) [MESH\\_NUMBER\\_OF\\_COMPONENTS\\_GET](#) (MESH, ERR, ERROR)
 

*Gets the number of mesh components for a mesh identified by a pointer.*
- subroutine [MESH\\_NUMBER\\_OF\\_COMPONENTS\\_SET\\_NUMBER](#) (USER\_NUMBER, REGION, NUMBER\_OF\_COMPONENTS, ERR, ERROR,\*)
 

*Changes/sets the number of mesh components for a mesh identified by a given user number on a region.*
- subroutine [MESH\\_NUMBER\\_OF\\_COMPONENTS\\_SET\\_PTR](#) (MESH, NUMBER\_OF\_COMPONENTS, ERR, ERROR,\*)
 

*Changes/sets the number of mesh components for a mesh identified by a pointer.*
- INTEGER(INTG) [MESH\\_NUMBER\\_OF\\_ELEMENTS\\_GET](#) (MESH, ERR, ERROR)

*Gets the number of elements for a mesh identified by a pointer.*

- subroutine [MESH\\_NUMBER\\_OF\\_ELEMENTS\\_SET\\_NUMBER](#) (USER\_NUMBER, REGION, NUMBER\_OF\_ELEMENTS, ERR, ERROR,\*)

*Changes/sets the number of elements for a mesh identified by a given user number on a region.*

- subroutine [MESH\\_NUMBER\\_OF\\_ELEMENTS\\_SET\\_PTR](#) (MESH, NUMBER\_OF\_ELEMENTS, ERR, ERROR,\*)

*Changes/sets the number of elements for a mesh identified by a pointer.*

- subroutine [MESH\\_TOPOLOGY\\_CALCULATE](#) (TOPOLOGY, ERR, ERROR,\*)

*Calculates the mesh topology.*

- subroutine [MESH\\_TOPOLOGY\\_DOFS\\_CALCULATE](#) (TOPOLOGY, ERR, ERROR,\*)

*Calculates the degrees-of-freedom for a mesh topology.*

- subroutine [MESH\\_TOPOLOGY\\_DOFS\\_FINALISE](#) (TOPOLOGY, ERR, ERROR,\*)

*Finalises the dof data structures for a mesh topology and deallocates any memory.*

- subroutine [MESH\\_TOPOLOGY\\_DOFS\\_INITIALISE](#) (TOPOLOGY, ERR, ERROR,\*)

*Initialises the dofs in a given mesh topology.*

- subroutine [MESH\\_TOPOLOGY\\_ELEMENTS\\_BASIS\\_SET](#) (GLOBAL\_NUMBER, ELEMENTS, BASIS, ERR, ERROR,\*)

*Changes/sets the basis for a global element identified by a given global number.*

- subroutine [MESH\\_TOPOLOGY\\_ELEMENTS\\_CREATE\\_FINISH](#) (MESH, MESH\_COMPONENT\_NUMBER, ERR, ERROR,\*)

*Finishes the process of creating elements for a specified mesh component in a mesh topology.*

- subroutine [MESH\\_TOPOLOGY\\_ELEMENTS\\_CREATE\\_START](#) (MESH, MESH\_COMPONENT\_NUMBER, BASIS, ELEMENTS, ERR, ERROR,\*)

*Starts the process of creating elements in the mesh component identified by MESH and component\_idx. The elements will be created with a default basis of BASIS. ELEMENTS is the returned pointer to the MESH\_ELEMENTS data structure.*

- subroutine [MESH\\_TOPOLOGY\\_ELEMENTS\\_DESTROY](#) (TOPOLOGY, ERR, ERROR,\*)

*Destroys the elements in a mesh topology.*

- subroutine [MESH\\_TOPOLOGY\\_ELEMENT\\_FINALISE](#) (ELEMENT, ERR, ERROR,\*)

*Finalises the given mesh topology element.*

- subroutine [MESH\\_TOPOLOGY\\_ELEMENT\\_INITIALISE](#) (ELEMENT, ERR, ERROR,\*)

*Initialises the given mesh topology element.*

- subroutine [MESH\\_TOPOLOGY\\_ELEMENTS\\_ELEMENT\\_BASIS\\_GET](#) (GLOBAL\_NUMBER, ELEMENTS, BASIS, ERR, ERROR,\*)

*Gets the basis for a mesh element identified by a given global number.*

- subroutine [MESH\\_TOPOLOGY\\_ELEMENTS\\_ELEMENT\\_BASIS\\_SET](#) (GLOBAL\_NUMBER, ELEMENTS, BASIS, ERR, ERROR,\*)

*Changes/sets the basis for a mesh element identified by a given global number.*

- subroutine [MESH\\_TOPOLOGY\\_ELEMENTS\\_ELEMENT\\_NODES\\_GET](#) (GLOBAL\_NUMBER, ELEMENTS, USER\_ELEMENT\_NODES, ERR, ERROR,\*)

*Gets the element nodes for a mesh element identified by a given global number.*

- subroutine [MESH\\_TOPOLOGY\\_ELEMENTS\\_ELEMENT\\_NODES\\_SET](#) (GLOBAL\_NUMBER, ELEMENTS, USER\_ELEMENT\_NODES, ERR, ERROR,\*)

*Changes/sets the element nodes for a mesh element identified by a given global number.*

- subroutine [MESH\\_TOPOLOGY\\_ELEMENTS\\_ADJACENT\\_ELEMENTS\\_CALCULATE](#) (TOPOLOGY, ERR, ERROR,\*)

*Calculates the element numbers surrounding an element in a mesh topology.*

- subroutine [MESH\\_TOPOLOGY\\_ELEMENTS\\_FINALISE](#) (TOPOLOGY, ERR, ERROR,\*)

*Finalises the elements data structures for a mesh topology and deallocates any memory.*

- subroutine [MESH\\_TOPOLOGY\\_ELEMENTS\\_INITIALISE](#) (TOPOLOGY, ERR, ERROR,\*)

*Initialises the elements in a given mesh topology.*

- subroutine [MESH\\_TOPOLOGY\\_ELEMENTS\\_NUMBER\\_GET](#) (GLOBAL\_NUMBER, USER\_NUMBER, ELEMENTS, ERR, ERROR,\*)

*Gets the user number for a global element identified by a given global number.*

- subroutine [MESH\\_TOPOLOGY\\_ELEMENTS\\_NUMBER\\_SET](#) (GLOBAL\_NUMBER, USER\_NUMBER, ELEMENTS, ERR, ERROR,\*)

*Changes/sets the user number for a global element identified by a given global number.*

- subroutine [MESH\\_TOPOLOGY\\_FINALISE](#) (MESH, ERR, ERROR,\*)

*Finalises the topology in the given mesh.*

- subroutine [MESH\\_TOPOLOGY\\_INITIALISE](#) (MESH, ERR, ERROR,\*)

*Initialises the topology for a given mesh.*

- subroutine [MESH\\_TOPOLOGY\\_NODE\\_FINALISE](#) (NODE, ERR, ERROR,\*)

*Finalises the given mesh topology node.*

- subroutine [MESH\\_TOPOLOGY\\_NODE\\_INITIALISE](#) (NODE, ERR, ERROR,\*)

*Initialises the given mesh topology node.*

- subroutine [MESH\\_TOPOLOGY\\_NODES\\_CALCULATE](#) (TOPOLOGY, ERR, ERROR,\*)

*Calculates the nodes used the mesh identified by a given mesh topology.*

- subroutine [MESH\\_TOPOLOGY\\_NODES\\_DERIVATIVES\\_CALCULATE](#) (TOPOLOGY, ERR, ERROR,\*)

*Calculates the number of derivatives at each node in a topology.*

- subroutine [MESH\\_TOPOLOGY\\_NODES\\_SURROUNDING\\_ELEMENTS\\_CALCULATE](#) (TOPOLOGY, ERR, ERROR,\*)

*Calculates the element numbers surrounding a node for a mesh.*

- subroutine **MESH\_TOPOLOGY\_NODES\_FINALISE** (TOPOLOGY, ERR, ERROR,\*)
 

*Finalises the nodes data structures for a mesh topology and deallocates any memory.*
- subroutine **MESH\_TOPOLOGY\_NODES\_INITIALISE** (TOPOLOGY, ERR, ERROR,\*)
 

*Initialises the nodes in a given mesh topology.*
- subroutine **MESH\_USER\_NUMBER\_FIND** (USER\_NUMBER, REGION, MESH, ERR, ERROR,\*)
 

*Finds and returns in MESH a pointer to the mesh identified by USER\_NUMBER in the given REGION. If no mesh with that number exists MESH is left nullified.*
- subroutine **MESHES\_FINALISE** (REGION, ERR, ERROR,\*)
 

*Finalises the meshes in the given region.*
- subroutine **MESHES\_INITIALISE** (REGION, ERR, ERROR,\*)
 

*Initialises the meshes for the given region.*

## Variables

- INTEGER(INTG), parameter **DECOMPOSITION\_ALL\_TYPE** = 1
 

*The decomposition contains all elements.*
- INTEGER(INTG), parameter **DECOMPOSITION\_CALCULATED\_TYPE** = 2
 

*The element decomposition is calculated by graph partitioning.*
- INTEGER(INTG), parameter **DECOMPOSITION\_USER\_DEFINED\_TYPE** = 3
 

*The user will set the element decomposition.*

### 6.35.1 Detailed Description

This module handles all mesh (node and element) routines.

### 6.35.2 Function Documentation

#### 6.35.2.1 subroutine **MESH\_ROUTINES::DECOMPOSITION\_CREATE\_FINISH** (TYPE(MESH\_TYPE),pointer **MESH**, TYPE(DECOMPOSITION\_TYPE),pointer **DECOMPOSITION**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

Finishes the creation of a domain decomposition on a given mesh.

#### Parameters:

**MESH** A pointer to the mesh to decompose.

**DECOMPOSITION** A pointer to the decomposition to finish creating

**ERR** The error code

**ERROR** The error string

Definition at line 115 of file mesh\_routines.f90.

References DECOMPOSITION\_ELEMENT\_DOMAIN\_CALCULATE(),  
 ROUTINES::DIAGNOSTIC\_OUTPUT\_TYPE, BASE\_ROUTINES::DIAGNOSTICS1, BASE\_  
 ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and  
 KINDS::PTR.

Referenced by FIELD\_IO\_ROUTINES::FIE().

Here is the call graph for this function:

Here is the caller graph for this function:

### 6.35.2.2 subroutine MESH\_ROUTINES::DECOMPOSITION\_CREATE\_START (INTEGER(INTG),intent(in) *USER\_NUMBER*, TYPE(MESH\_TYPE),pointer *MESH*, TYPE(DECOMPOSITION\_TYPE),pointer *DECOMPOSITION*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Starts the creation of a domain decomposition for a given mesh.

#### Parameters:

**USER\_NUMBER** The user number of the decomposition

**MESH** A pointer to the mesh to decompose

**DECOMPOSITION** On return a pointer to the created decomposition. Must not be associated on entry.

**ERR** The error code

**ERROR** The error string

Definition at line 185 of file mesh\_routines.f90.

References DECOMPOSITION\_ALL\_TYPE, BASE\_ROUTINES::ENTERS(), BASE\_  
 ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and KINDS::PTR.

Referenced by FIELD\_IO\_ROUTINES::FIE().

Here is the call graph for this function:

Here is the caller graph for this function:

### 6.35.2.3 subroutine MESH\_ROUTINES::DECOMPOSITION\_DESTROY (INTEGER(INTG),intent(in) *USER\_NUMBER*, TYPE(MESH\_TYPE),pointer *MESH*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Destroys a domain decomposition identified by a user number and deallocates all memory.

#### Parameters:

**USER\_NUMBER** The user number of the decomposition to destroy.

**MESH** A pointer to the mesh containing the decomposition.

**ERR** The error code

**ERROR** The error string

Definition at line 276 of file mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `KINDS::PTR`.

Referenced by `DECOMPOSITION_NUMBER_OF_DOMAINS_SET()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.4 subroutine MESH\_ROUTINES::DECOMPOSITION\_ELEMENT\_DOMAIN\_CALCULATE** (TYPE(DECOMPOSITION\_TYPE),pointer *DECOMPOSITION*,  
**INTEGER(INTG),intent(out) *ERR***, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)  
[private]

Calculates the element domains for a decomposition of a mesh.

#### Parameters:

***DECOMPOSITION*** A pointer to the decomposition to calculate the element domains for.

***ERR*** The error code

***ERROR*** The error string

Definition at line 364 of file mesh\_routines.f90.

References    `KINDS::SP`,    `COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT`,  
`COMP_ENVIRONMENT::COMPUTATIONAL_NODE_NUMBER_GET()`,    `COMP_ENVIRONMENT::COMPUTATIONAL_NODES_NUMBER_GET()`,    `DECOMPOSITION_ALL_TYPE`,    `DECOMPOSITION_CALCULATED_TYPE`,    `DECOMPOSITION_USER_DEFINED_TYPE`,    `BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE`,    `BASE_ROUTINES::DIAGNOSTICS1`,  
`BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,  
`KINDS::INTG`,    `CMISS_MPI::MPI_ERROR_CHECK()`,    `CMISS_PARMETIS::PARMETIS_PARTMESHKWAY()`, and `KINDS::PTR`.

Referenced by `DECOMPOSITION_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.5 subroutine MESH\_ROUTINES::DECOMPOSITION\_ELEMENT\_DOMAIN\_GET** (TYPE(DECOMPOSITION\_TYPE),pointer *DECOMPOSITION*,  
**INTEGER(INTG),intent(in) *GLOBAL\_ELEMENT\_NUMBER***,  
**INTEGER(INTG),intent(out) *DOMAIN\_NUMBER***, **INTEGER(INTG),intent(out) *ERR***,  
**TYPE(VARYING\_STRING),intent(out) *ERROR***, \*) [private]

Gets the domain for a given element in a decomposition of a mesh.

#### Parameters:

***DECOMPOSITION*** A pointer to the decomposition to set the element domain for

***GLOBAL\_ELEMENT\_NUMBER*** The global element number to set the domain for.

***DOMAIN\_NUMBER*** The domain of the global element.

***ERR*** The error code

**ERROR** The error string

Definition at line 554 of file mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `KINDS::PTR`.

Here is the call graph for this function:

**6.35.2.6 subroutine MESH\_ROUTINES::DECOMPOSITION\_ELEMENT\_DOMAIN\_SET**  
`(TYPE(DECOMPOSITION_TYPE),pointer DECOMPOSITION,`  
`INTEGER(INTG),intent(in) GLOBAL_ELEMENT_NUMBER,`  
`INTEGER(INTG),intent(in) DOMAIN_NUMBER, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Sets the domain for a given element in a decomposition of a mesh.

#### Parameters:

**DECOMPOSITION** A pointer to the decomposition to set the element domain for  
**GLOBAL\_ELEMENT\_NUMBER** The global element number to set the domain for.  
**DOMAIN\_NUMBER** The domain of the global element.  
**ERR** The error code  
**ERROR** The error string

Definition at line 610 of file mesh\_routines.f90.

References    `COMP_ENVIRONMENT::COMPUTATIONAL_NODES_NUMBER_GET()`,    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `KINDS::PTR`.

Here is the call graph for this function:

**6.35.2.7 subroutine MESH\_ROUTINES::DECOMPOSITION\_MESH\_COMPONENT\_NUMBER\_GET**  
`(TYPE(DECOMPOSITION_TYPE),pointer DECOMPOSITION,`  
`INTEGER(INTG),intent(out) MESH_COMPONENT_NUMBER,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`  
`[private]`

Gets the mesh component number which will be used for the decomposition of a mesh.

#### Parameters:

**DECOMPOSITION** A pointer to the decomposition to get the mesh component for  
**MESH\_COMPONENT\_NUMBER** The mesh component number to get  
**ERR** The error code  
**ERROR** The error string

Definition at line 677 of file mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

---

**6.35.2.8 subroutine MESH\_ROUTINES::DECOMPOSITION\_MESH\_COMPONENT\_NUMBER\_SET** (TYPE(DECOMPOSITION\_TYPE),pointer *DECOMPOSITION*,  
**INTEGER(INTG),intent(in) MESH\_COMPONENT\_NUMBER,**  
**INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**  
**[private]**

Sets/changes the mesh component number which will be used for the decomposition of a mesh.

**Parameters:**

*DECOMPOSITION* A pointer to the decomposition to set the mesh component for

*MESH\_COMPONENT\_NUMBER* The mesh component number to set

*ERR* The error code

*ERROR* The error string

Definition at line 715 of file mesh\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.35.2.9 subroutine MESH\_ROUTINES::DECOMPOSITION\_NUMBER\_OF\_DOMAINS\_GET** (TYPE(DECOMPOSITION\_TYPE),pointer *DECOMPOSITION*,  
**INTEGER(INTG),intent(out) NUMBER\_OF\_DOMAINS, INTEGER(INTG),intent(out)**  
**ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)** [private]

Gets the number of domains for a decomposition.

**Parameters:**

*DECOMPOSITION* A pointer to the decomposition to get the number of domains for.

*NUMBER\_OF\_DOMAINS* The number of domains to get.

*ERR* The error code

*ERROR* The error string

Definition at line 762 of file mesh\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.35.2.10 subroutine MESH\_ROUTINES::DECOMPOSITION\_NUMBER\_OF\_DOMAINS\_SET** (TYPE(DECOMPOSITION\_TYPE),pointer *DECOMPOSITION*,  
**INTEGER(INTG),intent(in) NUMBER\_OF\_DOMAINS, INTEGER(INTG),intent(out)**  
**ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets/changes the number of domains for a decomposition.

**Parameters:**

*DECOMPOSITION* A pointer to the decomposition to set the number of domains for.

**NUMBER\_OF\_DOMAINS** The number of domains to set.

**ERR** The error code

**ERROR** The error string

Definition at line 796 of file mesh\_routines.f90.

References KINDS::DP, COMP\_ENVIRONMENT::COMPUTATIONAL\_NODE\_NUMBER\_GET(), COMP\_ENVIRONMENT::COMPUTATIONAL\_NODES\_NUMBER\_GET(), DECOMPOSITION\_ALL\_TYPE, DECOMPOSITION\_CALCULATED\_TYPE, DECOMPOSITION\_DESTROY(), DECOMPOSITION\_USER\_DEFINED\_TYPE, BASE\_ROUTINES::DIAGNOSTIC\_OUTPUT\_TYPE, BASE\_ROUTINES::DIAGNOSTICS1, BASE\_ROUTINES::DIAGNOSTICS2, DOMAIN\_MAPPINGS::DOMAIN\_LOCAL\_BOUNDARY, DOMAIN\_MAPPINGS::DOMAIN\_LOCAL\_GHOST, DOMAIN\_MAPPINGS::DOMAIN\_LOCAL\_INTERNAL, DOMAIN\_MAPPINGS::DOMAIN\_LOCAL\_FROM\_GLOBAL\_CALCULATE(), DOMAIN\_MAPPINGS::DOMAIN\_MAPPING\_FINALISE(), DOMAIN\_MAPPINGS::DOMAIN\_MAPPINGS\_MAPPING\_GLOBAL\_INITIALISE(), DOMAIN\_MAPPINGS::DOMAIN\_MAPPINGS\_MAPPING\_INITIALISE(), DOMAIN\_MAPPINGS\_NODES\_FINALISE(), DOMAIN\_MAPPINGS\_NODES\_INITIALISE(), DOMAIN\_TOPOLOGY\_FINALISE(), DOMAIN\_TOPOLOGY\_INITIALISE(), DOMAIN\_TOPOLOGY\_LINE\_INITIALISE(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), KINDS::INTG, LISTS::LIST\_CREATE\_FINISH(), LISTS::LIST\_CREATE\_START(), LISTS::LIST\_DATA\_TYPE\_SET(), LISTS::LIST\_DESTROY(), LISTS::LIST\_INITIAL\_SIZE\_SET(), LISTS::LIST\_INTG\_TYPE, LISTS::LIST\_REMOVE\_DUPLICATES(), and KINDS::PTR.

Referenced by FIELD\_IO\_ROUTINES::FIE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.11 subroutine MESH\_ROUTINES::DOMAIN\_MAPPINGS\_NODES\_FINALISE**  
**(TYPE(DOMAIN\_MAPPINGS\_TYPE),pointer DOMAIN\_MAPPINGS,**  
**INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,**  
 $\ast$ ) [private]

Finalises the node mapping in the given domain mappings.

### **Todo**

pass in the nodes mapping

### **Parameters:**

**DOMAIN\_MAPPINGS** A pointer to the domain mapping to finalise the nodes for

**ERR** The error code

**ERROR** The error string

Definition at line 3179 of file mesh\_routines.f90.

References DOMAIN\_MAPPINGS::DOMAIN\_MAPPINGS\_MAPPING\_FINALISE(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by DECOMPOSITION\_NUMBER\_OF\_DOMAINS\_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.35.2.12 subroutine MESH\_ROUTINES::DOMAIN\_MAPPINGS\_NODES\_INITIALISE**  
 (**TYPE(DOMAIN\_MAPPINGS\_TYPE)**,pointer **DOMAIN\_MAPPINGS**,  
**INTEGER(INTG),intent(out)** **ERR**, **TYPE(VARYING\_STRING),intent(out)** **ERROR**,  
 \*) [private]

Initialises the node mapping in the given domain mapping.

#### Todo

finalise on error

#### Parameters:

**DOMAIN\_MAPPINGS** A pointer to the domain mappings to initialise the nodes for  
**ERR** The error code  
**ERROR** The error string

Definition at line 3211 of file mesh\_routines.f90.

References DOMAIN\_MAPPINGS::DOMAIN\_MAPPINGS\_MAPPING\_INITIALISE(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by DECOMPOSITION\_NUMBER\_OF\_DOMAINS\_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.13 subroutine MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_CALCULATE**  
 (**TYPE(DOMAIN\_TOPOLOGY\_TYPE)**,pointer **TOPOLOGY**,  
**INTEGER(INTG),intent(out)** **ERR**, **TYPE(VARYING\_STRING),intent(out)** **ERROR**,  
 \*) [private]

Calculates the domain topology.

#### Parameters:

**TOPOLOGY** A pointer to the domain topology to calculate.  
**ERR** The error code  
**ERROR** The error string

Definition at line 3247 of file mesh\_routines.f90.

References DOMAIN\_TOPOLOGY\_NODES\_SURROUNDING\_ELEMENTS\_CALCULATE(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by DOMAIN\_TOPOLOGY\_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.14 subroutine MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_DOFS\_FINALISE**  
 (**TYPE(DOMAIN\_TOPOLOGY\_TYPE)**,pointer **TOPOLOGY**,  
**INTEGER(INTG),intent(out)** **ERR**, **TYPE(VARYING\_STRING),intent(out)** **ERROR**,  
 \*) [private]

Finalises the dofs in the given domain topology.

**Todo**

pass in the dofs topology

**Parameters:**

**TOPOLOGY** A pointer to the domain topology to finalise the dofs for

**ERR** The error code

**ERROR** The error string

Definition at line 3466 of file mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `DOMAIN_TOPOLOGY_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.15 subroutine MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_DOFS\_INITIALISE  
(TYPE(DOMAIN\_TOPOLOGY\_TYPE),pointer TOPOLOGY,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*) [private]**

Initialises the dofs data structures for a domain topology.

**Todo**

finalise on exit

**Parameters:**

**TOPOLOGY** A pointer to the domain topology to initialise the dofs for

**ERR** The error code

**ERROR** The error string

Definition at line 3498 of file mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `DOMAIN_TOPOLOGY_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.16 subroutine MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_ELEMENT\_FINALISE  
(TYPE(DOMAIN\_ELEMENT\_TYPE) ELEMENT, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Finalises the given domain topology element.

**Parameters:**

**ELEMENT** The domain element to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 3534 of file mesh\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `DOMAIN_TOPOLOGY_ELEMENTS_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.35.2.17 subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENT_INITIALISE`(`TYPE(DOMAIN_ELEMENT_TYPE)` *ELEMENT*, `INTEGER(INTG)`,`intent(out)` *ERR*, `TYPE(VARYING_STRING)`,`intent(out)` *ERROR*, \*) [private]

Initialises the given domain topology element.

##### Parameters:

**ELEMENT** The domain element to initialise

**ERR** The error code

**ERROR** The error string

Definition at line 3559 of file mesh\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `DOMAIN_TOPOLOGY_INITIALISE_FROM_MESH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.35.2.18 subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_FINALISE`(`TYPE(DOMAIN_TOPOLOGY_TYPE)`,pointer *TOPOLOGY*, `INTEGER(INTG)`,`intent(out)` *ERR*, `TYPE(VARYING_STRING)`,`intent(out)` *ERROR*, \*) [private]

Finalises the elements in the given domain topology.

##### Todo

pass in the domain elements

##### Parameters:

**TOPOLOGY** A pointer to the domain topology to finalise the elements for

**ERR** The error code

**ERROR** The error string

Definition at line 3585 of file mesh\_routines.f90.

References DOMAIN\_TOPOLOGY\_ELEMENT\_FINALISE(),  
 BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().  
 BASE\_ROUTINES::ENTERS(),

Referenced by DOMAIN\_TOPOLOGY\_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.19 subroutine MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_ELEMENTS\_INITIALISE** (TYPE(DOMAIN\_TOPOLOGY\_TYPE),pointer *TOPOLOGY*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Initialises the element data structures for a domain topology.

### Todo

finalise on error

#### Parameters:

***TOPOLOGY*** A pointer to the domain topology to initialise the elements for

***ERR*** The error code

***ERROR*** The error string

Definition at line 3621 of file mesh\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by DOMAIN\_TOPOLOGY\_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.20 subroutine MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_FINALISE**  
 (TYPE(DOMAIN\_TYPE),pointer *DOMAIN*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Finalises the topology in the given domain.

### Todo

pass in domain topology

#### Parameters:

***DOMAIN*** A pointer to the domain to finalise the topology for

***ERR*** The error code

***ERROR*** The error string

Definition at line 3659 of file mesh\_routines.f90.

References DOMAIN\_TOPOLOGY\_DOFS\_FINALISE(), DOMAIN\_TOPOLOGY\_ELEMENTS\_FINALISE(), DOMAIN\_TOPOLOGY\_LINES\_FINALISE(), DOMAIN\_TOPOLOGY\_NODES\_FINALISE(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by DECOMPOSITION\_NUMBER\_OF\_DOMAINS\_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.35.2.21 subroutine MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_INITIALISE (*TYPE(DOMAIN\_TYPE)*,pointer *DOMAIN*, INTEGER(*INTG*),intent(out) *ERR*, *TYPE(VARYING\_STRING)*,intent(out) *ERROR*, \*) [private]

Initialises the topology for a given domain.

##### **Todo**

finalise on error

##### **Parameters:**

***ERR*** The error code

***ERROR*** The error string

Definition at line 3692 of file mesh\_routines.f90.

References DOMAIN\_TOPOLOGY\_CALCULATE(), DOMAIN\_TOPOLOGY\_DOFS\_INITIALISE(), DOMAIN\_TOPOLOGY\_ELEMENTS\_INITIALISE(), DOMAIN\_TOPOLOGY\_INITIALISE\_FROM\_MESH(), DOMAIN\_TOPOLOGY\_LINES\_INITIALISE(), DOMAIN\_TOPOLOGY\_NODES\_INITIALISE(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by DECOMPOSITION\_NUMBER\_OF\_DOMAINS\_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.35.2.22 subroutine MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_INITIALISE\_FROM\_MESH (*TYPE(DOMAIN\_TYPE)*,pointer *DOMAIN*, INTEGER(*INTG*),intent(out) *ERR*, *TYPE(VARYING\_STRING)*,intent(out) *ERROR*, \*) [private]

Initialises the local domain topology from the mesh topology.

##### **Parameters:**

***DOMAIN*** A pointer to the domain to initialise the domain topology from the mesh topology for

***ERR*** The error code

***ERROR*** The error string

Definition at line 3293 of file mesh\_routines.f90.

References BASE\_ROUTINES::DIAGNOSTIC\_OUTPUT\_TYPE, BASE\_ROUTINES::DIAGNOSTICS1, DOMAIN\_TOPOLOGY\_ELEMENT\_INITIALISE(), DOMAIN\_TOPOLOGY\_NODE\_INITIALISE(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and KINDS::PTR.

Referenced by DOMAIN\_TOPOLOGY\_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.23 subroutine MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_LINE\_FINALISE  
(TYPE(DOMAIN\_LINE\_TYPE) *LINE*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Finalises a line in the given domain topology and deallocates all memory.

**Parameters:**

*LINE* The domain line to finalise

*ERR* The error code

*ERROR* The error string

Definition at line 3740 of file mesh\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by DOMAIN\_TOPOLOGY\_LINES\_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.24 subroutine MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_LINE\_INITIALISE  
(TYPE(DOMAIN\_LINE\_TYPE) *LINE*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Initialises the line data structure for a domain topology line.

**Parameters:**

*LINE* The domain line to initialise

*ERR* The error code

*ERROR* The error string

Definition at line 3768 of file mesh\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by DECOMPOSITION\_NUMBER\_OF\_DOMAINS\_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.25 subroutine MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_LINES\_FINALISE  
(TYPE(DOMAIN\_TOPOLOGY\_TYPE),pointer *TOPOLOGY*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
\*) [private]**

Finalises the lines in the given domain topology.

**Todo**

pass in domain lines

**Parameters:**

**TOPOLOGY** A pointer to the domain topology to finalise the lines for

**ERR** The error code

**ERROR** The error string

Definition at line 3793 of file mesh\_routines.f90.

References DOMAIN\_TOPOLOGY\_LINE\_FINALISE(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by DOMAIN\_TOPOLOGY\_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.26 subroutine MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_LINES\_INITIALISE  
(TYPE(DOMAIN\_TOPOLOGY\_TYPE),pointer TOPOLOGY,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*) [private]**

Initialises the line data structures for a domain topology.

**Todo**

finalise on error

**Parameters:**

**TOPOLOGY** A pointer to the domain topology to initialise the lines for

**ERR** The error code

**ERROR** The error string

Definition at line 3829 of file mesh\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by DOMAIN\_TOPOLOGY\_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.27 subroutine MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_NODE\_FINALISE  
(TYPE(DOMAIN\_NODE\_TYPE) NODE, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Finalises the given domain topology node and deallocates all memory.

**Parameters:**

**NODE** The domain node to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 3864 of file mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `DOMAIN_TOPOLOGY_NODES_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.35.2.28 subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_NODE_INITIALISE` `(TYPE(DOMAIN_NODE_TYPE) NODE, INTEGER(INTG),intent(out) ERR,` `TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Initialises the given domain topology node.

##### Parameters:

**NODE** The domain node to initialise

**ERR** The error code

**ERROR** The error string

Definition at line 3891 of file mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `DOMAIN_TOPOLOGY_INITIALISE_FROM_MESH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.35.2.29 subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_FINALISE` `(TYPE(DOMAIN_TOPOLOGY_TYPE),pointer TOPOLOGY,` `INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,` `*) [private]`

Finalises the nodees in the given domain topology.

##### Todo

pass in domain nodes

##### Parameters:

**TOPOLOGY** A pointer to the domain topology to initialise the nodes for

**ERR** The error code

**ERROR** The error string

Definition at line 3920 of file mesh\_routines.f90.

References DOMAIN\_TOPOLOGY\_NODE\_FINALISE(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by DOMAIN\_TOPOLOGY\_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.30 subroutine MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_NODES\_INITIALISE  
(TYPE(DOMAIN\_TOPOLOGY\_TYPE),pointer *TOPOLOGY*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
\*) [private]**

Initialises the nodes data structures for a domain topology.

#### **Todo**

finalise on error

#### **Parameters:**

***TOPOLOGY*** A pointer to the domain topology to initialise the nodes for

***ERR*** The error code

***ERROR*** The error string

Definition at line 3956 of file mesh\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by DOMAIN\_TOPOLOGY\_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.31 subroutine MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_NODES\_SURROUNDING\_ELEMENTS\_CALCULATE (TYPE(DOMAIN\_TOPOLOGY\_TYPE),pointer  
*TOPOLOGY*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Calculates the element numbers surrounding a node for a domain.

#### **Parameters:**

***TOPOLOGY*** A pointer to the domain topology to calculate the elements surrounding the nodes for

***ERR*** The error code

***ERROR*** The error string

Definition at line 3994 of file mesh\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by DOMAIN\_TOPOLOGY\_CALCULATE().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.35.2.32 subroutine MESH\_ROUTINES::MESH\_CREATE\_FINISH (TYPE(REGION\_TYPE),pointer *REGION*, TYPE(MESH\_TYPE),pointer *MESH*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Finishes the process of creating a mesh on a region.

#### **Todo**

remove region

#### **Parameters:**

***REGION*** A pointer to the region containing the mesh  
***MESH*** A pointer to the mesh to finish creating  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 4083 of file mesh\_routines.f90.

References                    BASE\_ROUTINES::DIAGNOSTIC\_OUTPUT\_TYPE,                    BASE\_ROUTINES::DIAGNOSTICS1,    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_ROUTINES::EXITS(),    MESH\_TOPOLOGY\_CALCULATE(), and KINDS::PTR.

Referenced by FIELD\_IO\_ROUTINES::FIELD\_IO\_IMPORT\_GLOBAL\_MESH(), and GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_REGULAR\_CREATE\_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.35.2.33 subroutine MESH\_ROUTINES::MESH\_CREATE\_START (INTEGER(INTG),intent(in) *USER\_NUMBER*, TYPE(REGION\_TYPE),pointer *REGION*, INTEGER(INTG),intent(in) *NUMBER\_OF\_DIMENSIONS*, TYPE(MESH\_TYPE),pointer *MESH*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Starts the process of creating a mesh defined by a user number with the specified NUMBER\_OF\_DIMENSIONS in the region identified by REGION.

#### **Parameters:**

***USER\_NUMBER*** The user number of the mesh to create  
***REGION*** A pointer to the region to create the mesh on  
***NUMBER\_OF\_DIMENSIONS*** The number of dimensions in the mesh.  
***MESH*** On exit, a pointer to the created mesh. Must not be associated on entry.  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 4354 of file mesh\_routines.f90.

References                    BASE\_ROUTINES::ENTERS(),                    BASE\_ROUTINES::ERRORS(),                    BASE\_ROUTINES::EXITS(),    MESH\_INITIALISE(),    MESH\_TOPOLOGY\_INITIALISE(),    MESH\_USER\_NUMBER\_FIND(), and KINDS::PTR.

Referenced by FIELD\_IO\_ROUTINES::FIELD\_IO\_IMPORT\_GLOBAL\_MESH(), and GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_REGULAR\_CREATE\_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.34 subroutine MESH\_ROUTINES::MESH\_DESTROY (INTEGER(INTG),intent(in) *USER\_NUMBER*, TYPE(REGION\_TYPE),pointer *REGION*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Destroys the mesh identified by a user number on the given region and deallocates all memory.

**Parameters:**

***USER\_NUMBER*** The user number of the mesh to destroy

***REGION*** A pointer to the region containing the mesh

***ERR*** The error code

***ERROR*** The error string

Definition at line 4450 of file mesh\_routines.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_ROUTINES::EXITS(), MESH\_FINALISE(), and KINDS::PTR.

Referenced by MESHES\_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.35 subroutine MESH\_ROUTINES::MESH\_FINALISE (TYPE(MESH\_TYPE),pointer *MESH*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Finalises a mesh and deallocates all memory.

**Parameters:**

***MESH*** A pointer to the mesh to finalise

***ERR*** The error code

***ERROR*** The error string

Definition at line 4534 of file mesh\_routines.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_ROUTINES::EXITS(), and MESH\_TOPOLOGY\_FINALISE().

Referenced by MESH\_DESTROY().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.35.2.36 subroutine MESH\_ROUTINES::MESH\_INITIALISE (TYPE(MESH\_TYPE),pointer MESH, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Initialises a mesh.

**Parameters:**

**MESH** A pointer to the mesh to initialise

**ERR** The error code

**ERROR** The error string

Definition at line 4563 of file mesh\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Referenced by MESH\_CREATE\_START().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.37 INTEGER(INTG) MESH\_ROUTINES::MESH\_NUMBER\_OF\_COMPONENTS\_GET (TYPE(MESH\_TYPE),pointer MESH, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

Gets the number of mesh components for a mesh identified by a pointer.

**Parameters:**

**MESH** A pointer to the mesh to set the number of components for

**ERR** The error code

**ERROR** The error string

Definition at line 4608 of file mesh\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.35.2.38 subroutine MESH\_ROUTINES::MESH\_NUMBER\_OF\_COMPONENTS\_SET\_NUMBER (INTEGER(INTG),intent(in) USER\_NUMBER, TYPE(REGION\_TYPE),pointer REGION, INTEGER(INTG),intent(in) NUMBER\_OF\_COMPONENTS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Changes/sets the number of mesh components for a mesh identified by a given user number on a region.

**Parameters:**

**USER\_NUMBER** The user number of the mesh to set the number of components for

**REGION** A pointer to the region containing the mesh

**NUMBER\_OF\_COMPONENTS** The number of components to set for the mesh

**ERR** The error code

**ERROR** The error string

Definition at line 4639 of file mesh\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MESH_NUMBER_OF_COMPONENTS_SET_PTR()`, and `MESH_USER_NUMBER_FIND()`.

Here is the call graph for this function:

```
6.35.2.39 subroutine MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET_PTR
 (TYPE(MESH_TYPE),pointer MESH, INTEGER(INTG),intent(in)
 NUMBER_OF_COMPONENTS, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Changes/sets the number of mesh components for a mesh identified by a pointer.

#### Parameters:

**MESH** A pointer to the mesh to set the number of components for

**NUMBER\_OF\_COMPONENTS** The number of components to set.

**ERR** The error code

**ERROR** The error string

Definition at line 4677 of file mesh\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MESH_TOPOLOGY_ELEMENTS_INITIALISE()`, `MESH_TOPOLOGY_NODES_INITIALISE()`, and `KINDS::PTR`.

Referenced by `MESH_NUMBER_OF_COMPONENTS_SET_NUMBER()`.

Here is the call graph for this function:

Here is the caller graph for this function:

```
6.35.2.40 INTEGER(INTG) MESH_ROUTINES::MESH_NUMBER_OF_ELEMENTS_GET
 (TYPE(MESH_TYPE),pointer MESH, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR) [private]
```

Gets the number of elements for a mesh identified by a pointer.

#### Parameters:

**MESH** A pointer to the mesh to get the number of elements for

**ERR** The error code

**ERROR** The error string

Definition at line 4749 of file mesh\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

---

**6.35.2.41 subroutine MESH\_ROUTINES::MESH\_NUMBER\_OF\_ELEMENTS\_SET\_NUMBER**  
 (INTEGER(INTG),intent(in) *USER\_NUMBER*, TYPE(REGION\_TYPE),pointer  
*REGION*, INTEGER(INTG),intent(in) *NUMBER\_OF\_ELEMENTS*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Changes/sets the number of elements for a mesh identified by a given user number on a region.

**Parameters:**

***USER\_NUMBER*** The user number of the mesh to set the number of elements for  
***REGION*** A pointer to the region containing the mesh  
***NUMBER\_OF\_ELEMENTS*** The number of elements to set  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 4783 of file mesh\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(),    MESH\_NUMBER\_OF\_ELEMENTS\_SET\_PTR(),    and    MESH\_USER\_-NUMBER\_FIND().

Here is the call graph for this function:

**6.35.2.42 subroutine MESH\_ROUTINES::MESH\_NUMBER\_OF\_ELEMENTS\_SET\_PTR**  
 (TYPE(MESH\_TYPE),pointer *MESH*, INTEGER(INTG),intent(in)  
*NUMBER\_OF\_ELEMENTS*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Changes/sets the number of elements for a mesh identified by a pointer.

**Parameters:**

***MESH*** A pointer to the mesh to set the number of elements for  
***NUMBER\_OF\_ELEMENTS*** The number of elements to set  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 4821 of file mesh\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), and KINDS::PTR.

Referenced by MESH\_NUMBER\_OF\_ELEMENTS\_SET\_NUMBER().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.43 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_CALCULATE** (TYPE(MESH\_-TOPOLOGY\_TYPE),pointer *TOPOLOGY*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Calculates the mesh topology.

**Parameters:**

**TOPOLOGY** A pointer to the mesh topology to calculate  
**ERR** The error code  
**ERROR** The error string

Definition at line 4881 of file mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `MESH_TOPOLOGY_DOFS_CALCULATE()`,    `MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE()`,    `MESH_TOPOLOGY_NODES_CALCULATE()`,    `MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE()`,    and    `MESH_TOPOLOGY_NODES_SURROUNDING_ELEMENTS_CALCULATE()`.

Referenced by `MESH_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.44 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_DOFS\_-  
CALCULATE (TYPE(MESH\_TOPOLOGY\_TYPE),pointer TOPOLOGY,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*) [private]**

Calculates the degrees-of-freedom for a mesh topology.

**Parameters:**

**TOPOLOGY** A pointer to the mesh topology to calculate the dofs for  
**ERR** The error code  
**ERROR** The error string

Definition at line 4918 of file mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `MESH_TOPOLOGY_CALCULATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.45 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_DOFS\_-  
FINALISE (TYPE(MESH\_TOPOLOGY\_TYPE),pointer TOPOLOGY,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*) [private]**

Finalises the dof data structures for a mesh topology and deallocates any memory.

**Todo**

pass in dofs

**Parameters:**

**TOPOLOGY** A pointer to the mesh topology to finalise the dofs for

**ERR** The error code

**ERROR** The error string

Definition at line 4964 of file mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `MESH_TOPOLOGY_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.46 subroutine `MESH_ROUTINES::MESH_TOPOLOGY_DOFS_INITIALISE`** (TYPE(`MESH_TOPOLOGY_TYPE`),pointer *TOPOLOGY*,  
*INTEGER(INTG),intent(out)* *ERR*, *TYPE(VARYING\_STRING),intent(out)* *ERROR*,  
*\*) [private]*

Initialises the dofs in a given mesh topology.

#### **Todo**

finalise on error

#### **Parameters:**

***TOPOLOGY*** A pointer to the mesh topology to initialise the dofs for

**ERR** The error code

**ERROR** The error string

Definition at line 4994 of file mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `MESH_TOPOLOGY_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.47 subroutine `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENT_FINALISE`** (TYPE(`MESH_ELEMENT_TYPE`) *ELEMENT*, *INTEGER(INTG),intent(out)* *ERR*,  
*TYPE(VARYING\_STRING),intent(out)* *ERROR*, *\*) [private]*

Finalises the given mesh topology element.

#### **Parameters:**

***ELEMENT*** The mesh element to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 5285 of file mesh\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `MESH_TOPOLOGY_ELEMENTS_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### **6.35.2.48 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_ELEMENT\_INITIALISE (TYPE(MESH\_ELEMENT\_TYPE) ELEMENT, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Initialises the given mesh topology element.

**Parameters:**

**ELEMENT** The mesh element to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 5312 of file `mesh_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `MESH_TOPOLOGY_ELEMENTS_CREATE_START()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### **6.35.2.49 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_ELEMENTS\_ADJACENT\_- ELEMENTS\_CALCULATE (TYPE(MESH\_TOPOLOGY\_TYPE),pointer TOPOLOGY, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Calculates the element numbers surrounding an element in a mesh topology.

**Parameters:**

**TOPOLOGY** A pointer to the mesh topology to calculate the elements adjacent to elements for

**ERR** The error code

**ERROR** The error string

Definition at line 5603 of file `mesh_routines.f90`.

References `BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE`, `BASE_ROUTINES::DIAGNOSTICS1`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `LISTS::LIST_CREATE_FINISH()`, `LISTS::LIST_CREATE_START()`, `LISTS::LIST_DATA_TYPE_SET()`, `LISTS::LIST_DESTROY()`, `LISTS::LIST_INITIAL_SIZE_SET()`, `LISTS::LIST_INTG_TYPE`, `LISTS::LIST_REMOVE_DUPLICATES()`, and `KINDS::PTR`.

Referenced by `MESH_TOPOLOGY_CALCULATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.35.2.50 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_ELEMENTS\_BASIS\_SET**  
 (INTEGER(INTG),intent(in) *GLOBAL\_NUMBER*, TYPE(MESH\_-  
*ELEMENTS\_TYPE*),pointer *ELEMENTS*, TYPE(BASIS\_TYPE),pointer *BASIS*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Changes/sets the basis for a global element identified by a given global number.

#### **Todo**

use user number.

#### **Todo**

get method?

#### **Parameters:**

***GLOBAL\_NUMBER*** The global number of the element to set the basis for  
***ELEMENTS*** A pointer to the elements to set the basis for before element number?  
***BASIS*** A pointer to the basis to set. allow number version as well.  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 5029 of file mesh\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.35.2.51 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_ELEMENTS\_CREATE\_-  
 FINISH (TYPE(MESH\_TYPE),pointer *MESH*, INTEGER(INTG),intent(in)  
*MESH\_COMPONENT\_NUMBER*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Finishes the process of creating elements for a specified mesh component in a mesh topology.

#### **Parameters:**

***MESH*** A pointer to the mesh to finish creating the elements for  
***MESH\_COMPONENT\_NUMBER*** The mesh component number to finish creating the elements for  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 5084 of file mesh\_routines.f90.

References BASE\_ROUTINES::DIAGNOSTIC\_OUTPUT\_TYPE, BASE\_-ROUTINES::DIAGNOSTICS1, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and KINDS::PTR.

Referenced by FIELD\_IO\_ROUTINES::FIELD\_IO\_IMPORT\_GLOBAL\_MESH(), and GENERATED\_-MESH\_ROUTINES::GENERATED\_MESH\_REGULAR\_CREATE\_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.35.2.52 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_ELEMENTS\_CREATE\_START**  
**(TYPE(MESH\_TYPE),pointer MESH, INTEGER(INTG),intent(in)**  
**MESH\_COMPONENT\_NUMBER, TYPE(BASIS\_TYPE),pointer**  
**BASIS, TYPE(MESH\_ELEMENTS\_TYPE),pointer ELEMENTS,**  
**INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,**  
 $\ast$ )

Starts the process of creating elements in the mesh component identified by MESH and component\_idx. The elements will be created with a default basis of BASIS. ELEMENTS is the returned pointer to the MESH\_ELEMENTS data structure.

**Parameters:**

**MESH** A pointer to the mesh to start creating the elements on  
**MESH\_COMPONENT\_NUMBER** The mesh component number  
**BASIS** A pointer to the default basis to use  
**ELEMENTS** On return, a pointer to the created mesh elements  
**ERR** The error code  
**ERROR** The error string

Definition at line 5163 of file mesh\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_ROUTINES::EXITS(),    MESH\_TOPOLOGY\_ELEMENT\_INITIALISE(),    MESH\_TOPOLOGY\_ELEMENTS\_FINALISE(), and KINDS::PTR.

Referenced by FIELD\_IO\_ROUTINES::FIELD\_IO\_IMPORT\_GLOBAL\_MESH(), and GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_REGULAR\_CREATE\_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.53 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_ELEMENTS\_DESTROY**  
**(TYPE(MESH\_TOPOLOGY\_TYPE),pointer TOPOLOGY,**  
**INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,**  
 $\ast$ ) [private]

Destroys the elements in a mesh topology.

**Todo**

as this is a user routine it should take a mesh pointer like create start and finish? Split this into destroy and finalise?

**Parameters:**

**TOPOLOGY** A pointer to the mesh topology to destroy the elements for  
**ERR** The error code  
**ERROR** The error string

Definition at line 5241 of file mesh\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

---

**6.35.2.54 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_ELEMENTS\_-ELEMENT\_BASIS\_GET (INTEGER(INTG),intent(in) *GLOBAL\_NUMBER*, TYPE(MESH\_ELEMENTS\_TYPE),pointer *ELEMENTS*, TYPE(BASIS\_TYPE),pointer *BASIS*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Gets the basis for a mesh element identified by a given global number.

#### **Todo**

should take user number

#### **Parameters:**

***GLOBAL\_NUMBER*** The global number of the element to get the basis for  
***ELEMENTS*** A pointer to the elements to get the basis for before number?  
***BASIS*** A pointer to the basis to get  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 5340 of file mesh\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    and    **BASE\_ROUTINES::EXITS()**.

Here is the call graph for this function:

**6.35.2.55 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_ELEMENTS\_-ELEMENT\_BASIS\_SET (INTEGER(INTG),intent(in) *GLOBAL\_NUMBER*, TYPE(MESH\_ELEMENTS\_TYPE),pointer *ELEMENTS*, TYPE(BASIS\_TYPE),pointer *BASIS*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Changes/sets the basis for a mesh element identified by a given global number.

#### **Todo**

should take user number

#### **Parameters:**

***GLOBAL\_NUMBER*** The global number of the element to set the basis for  
***ELEMENTS*** A pointer to the elements to set the basis for before number?  
***BASIS*** A pointer to the basis to set  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 5384 of file mesh\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    and    **BASE\_ROUTINES::EXITS()**.

Referenced by **FIELD\_IO\_ROUTINES::FIELD\_IO\_IMPORT\_GLOBAL\_MESH()**.

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.35.2.56 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_ELEMENTS\_-ELEMENT\_NODES\_GET (INTEGER(INTG),intent(in) GLOBAL\_NUMBER, TYPE(MESH\_ELEMENTS\_TYPE),pointer ELEMENTS, INTEGER(INTG),dimension(:,),pointer USER\_ELEMENT\_NODES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Gets the element nodes for a mesh element identified by a given global number.

#### **Todo**

specify by user number not global number

#### **Parameters:**

**GLOBAL\_NUMBER** The global number of the element to set the nodes for  
**ELEMENTS** A pointer to the elements to set before number?  
**USER\_ELEMENT\_NODES** USER\_ELEMENT\_NODES(i). USER\_ELEMENT\_NODES(i) is the i'th user node number for the element  
**ERR** The error code  
**ERROR** The error string

Definition at line 5462 of file mesh\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.35.2.57 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_ELEMENTS\_-ELEMENT\_NODES\_SET (INTEGER(INTG),intent(in) GLOBAL\_NUMBER, TYPE(MESH\_ELEMENTS\_TYPE),pointer ELEMENTS, INTEGER(INTG),dimension(:,),intent(in) USER\_ELEMENT\_NODES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Changes/sets the element nodes for a mesh element identified by a given global number.

#### **Todo**

specify by user number not global number

#### **Parameters:**

**GLOBAL\_NUMBER** The global number of the element to set the nodes for  
**ELEMENTS** A pointer to the elements to set before number?  
**USER\_ELEMENT\_NODES** USER\_ELEMENT\_NODES(i). USER\_ELEMENT\_NODES(i) is the i'th user node number for the element  
**ERR** The error code  
**ERROR** The error string

Definition at line 5506 of file mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `NODE_ROUTINES::NODE_CHECK_EXISTS()`.

Referenced by `FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH()`, and `GENERATED_MESH_ROUTINES::GENERATED_MESH_REGULAR_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.58 subroutine `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_FINALISE`** (`TYPE(MESH_TOPOLOGY_TYPE)`,pointer *TOPOLOGY*,  
`INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR`,  
`*) [private]`

Finalises the elements data structures for a mesh topology and deallocates any memory.

#### **Todo**

pass in elements

#### **Parameters:**

*TOPOLOGY* A pointer to the mesh topology to finalise the elements for

*ERR* The error code

*ERROR* The error string

Definition at line 5815 of file `mesh_routines.f90`.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `MESH_TOPOLOGY_ELEMENT_FINALISE()`.

Referenced by `MESH_TOPOLOGY_ELEMENTS_CREATE_START()`, and `MESH_TOPOLOGY_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.59 subroutine `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_INITIALISE`** (`TYPE(MESH_TOPOLOGY_TYPE)`,pointer *TOPOLOGY*,  
`INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR`,  
`*) [private]`

Initialises the elements in a given mesh topology.

#### **Todo**

finalise on error

#### **Parameters:**

*TOPOLOGY* A pointer to the mesh topology to initialise the elements for

*ERR* The error code

*ERROR* The error string

Definition at line 5849 of file mesh\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `MESH_NUMBER_OF_COMPONENTS_SET_PTR()`, and `MESH_TOPOLOGY_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.60 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_ELEMENTS\_NUMBER\_GET (INTEGER(INTG),intent(in) *GLOBAL\_NUMBER*, INTEGER(INTG),intent(out) *USER\_NUMBER*, TYPE(MESH\_ELEMENTS\_TYPE),pointer *ELEMENTS*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Gets the user number for a global element identified by a given global number.

### **Todo**

Check that the user number doesn't already exist.

#### **Parameters:**

***GLOBAL\_NUMBER*** The global number of the elements to get.

***USER\_NUMBER*** The user number of the element to get

***ELEMENTS*** A pointer to the elements to get the user number for This should be the first parameter.

***ERR*** The error code

***ERROR*** The error string

Definition at line 5887 of file mesh\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.35.2.61 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_ELEMENTS\_NUMBER\_SET (INTEGER(INTG),intent(in) *GLOBAL\_NUMBER*, INTEGER(INTG),intent(in) *USER\_NUMBER*, TYPE(MESH\_ELEMENTS\_TYPE),pointer *ELEMENTS*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Changes/sets the user number for a global element identified by a given global number.

### **Todo**

Check that the user number doesn't already exist.

#### **Parameters:**

***GLOBAL\_NUMBER*** The global number of the elements to set.

***USER\_NUMBER*** The user number of the element to set

**ELEMENTS** A pointer to the elements to set the user number for This should be the first parameter.

**ERR** The error code

**ERROR** The error string

Definition at line 5929 of file mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.62 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_FINALISE**  
`(TYPE(MESH_TYPE),pointer MESH, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Finalises the topology in the given mesh.

#### **Todo**

pass in the mesh topology

#### **Parameters:**

**MESH** A pointer to the mesh to finalise the topology for

**ERR** The error code

**ERROR** The error string

Definition at line 5971 of file mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `MESH_TOPOLOGY_DOFS_FINALISE()`, `MESH_TOPOLOGY_ELEMENTS_FINALISE()`, `MESH_TOPOLOGY_NODES_FINALISE()`, and `KINDS::PTR`.

Referenced by `MESH_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.63 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_INITIALISE**  
`(TYPE(MESH_TYPE),pointer MESH, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Initialises the topology for a given mesh.

#### **Todo**

finalise on error

#### **Parameters:**

**MESH** A pointer to the mesh to initialise the mesh topology for

**ERR** The error code

**ERROR** The error string

Definition at line 6007 of file mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `MESH_TOPOLOGY_DOFS_INITIALISE()`,    `MESH_TOPOLOGY_ELEMENTS_INITIALISE()`, `MESH_TOPOLOGY_NODES_INITIALISE()`, and `KINDS::PTR`.

Referenced by `MESH_CREATE_START()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.64 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_NODE\_FINALISE  
(TYPE(MESH\_NODE\_TYPE) NODE, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Finalises the given mesh topology node.

#### Parameters:

**NODE** The mesh node to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 6054 of file mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `MESH_TOPOLOGY_NODES_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.65 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_NODE\_INITIALISE  
(TYPE(MESH\_NODE\_TYPE) NODE, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Initialises the given mesh topology node.

#### Parameters:

**NODE** The mesh node to initialise

**ERR** The error code

**ERROR** The error string

Definition at line 6080 of file mesh\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `MESH_TOPOLOGY_NODES_CALCULATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.35.2.66 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_NODES\_-  
CALCULATE (TYPE(MESH\_TOPOLOGY\_TYPE),pointer *TOPOLOGY*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
\*) [private]**

Calculates the nodes used the mesh identified by a given mesh topology.

**Parameters:**

***TOPOLOGY*** A pointer to the mesh topology

***ERR*** The error code

***ERROR*** The error string

Definition at line 6108 of file mesh\_routines.f90.

References            BASE\_ROUTINES::DIAGNOSTIC\_OUTPUT\_TYPE,            BASE\_-ROUTINES::DIAGNOSTICS1,    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_ROUTINES::EXITS(),    MESH\_TOPOLOGY\_NODE\_INITIALISE(),    TREES::TREE\_CREATE\_-FINISH(),    TREES::TREE\_CREATE\_START(),    TREES::TREE\_DESTROY(),    TREES::TREE\_-DETACH\_AND\_DESTROY(),    TREES::TREE\_INSERT\_TYPE\_SET(),    TREES::TREE\_ITEM\_-INSERT(), and TREES::TREE\_NO\_DUPLICATES\_ALLOWED.

Referenced by MESH\_TOPOLOGY\_CALCULATE().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.35.2.67 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_NODES\_DERIVATIVES\_-  
CALCULATE (TYPE(MESH\_TOPOLOGY\_TYPE),pointer *TOPOLOGY*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
\*) [private]**

Calculates the number of derivatives at each node in a topology.

**Parameters:**

***TOPOLOGY*** A pointer to the mesh topology to calculate the derivates at each node for

***ERR*** The error code

***ERROR*** The error string

Definition at line 6212 of file mesh\_routines.f90.

References            BASE\_ROUTINES::DIAGNOSTIC\_OUTPUT\_TYPE,            BASE\_-ROUTINES::DIAGNOSTICS1,    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_ROUTINES::EXITS(),    LISTS::LIST\_CREATE\_FINISH(),    LISTS::LIST\_CREATE\_START(),    LISTS::LIST\_DATA\_TYPE\_SET(),    LISTS::LIST\_DESTROY(),    LISTS::LIST\_INITIAL\_SIZE\_SET(),    LISTS::LIST\_INTG\_TYPE, and LISTS::LIST\_REMOVE\_DUPLICATES().

Referenced by MESH\_TOPOLOGY\_CALCULATE().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.35.2.68 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_NODES\_-  
FINALISE (TYPE(MESH\_TOPOLOGY\_TYPE),pointer TOPOLOGY,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*) [private]**

Finalises the nodes data structures for a mesh topology and deallocates any memory.

### **Todo**

pass in nodes

#### **Parameters:**

**TOPOLOGY** A pointer to the mesh topology to finalise the nodes for  
**ERR** The error code  
**ERROR** The error string

Definition at line 6402 of file mesh\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    **BASE\_-ROUTINES::EXITS()**, and **MESH\_TOPOLOGY\_NODE\_FINALISE()**.

Referenced by **MESH\_TOPOLOGY\_FINALISE()**.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.69 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_NODES\_-  
INITIALISE (TYPE(MESH\_TOPOLOGY\_TYPE),pointer TOPOLOGY,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*) [private]**

Initialises the nodes in a given mesh topology.

### **Todo**

finalise on errors

#### **Parameters:**

**TOPOLOGY** A pointer to the mesh topology to initialise the nodes for  
**ERR** The error code  
**ERROR** The error string

Definition at line 6437 of file mesh\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    and    **BASE\_-ROUTINES::EXITS()**.

Referenced by **MESH\_NUMBER\_OF\_COMPONENTS\_SET\_PTR()**, and **MESH\_TOPOLOGY\_-INITIALISE()**.

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.35.2.70 subroutine MESH\_ROUTINES::MESH\_TOPOLOGY\_NODES\_SURROUNDING\_ELEMENTS\_CALCULATE** (*TYPE(MESH\_TOPOLOGY\_TYPE)*,*pointer TOPOLOGY*,  
*INTEGER(INTG),intent(out)* *ERR*, *TYPE(VARYING\_STRING),intent(out)* *ERROR*,  
*\*) [private]*

Calculates the element numbers surrounding a node for a mesh.

**Parameters:**

*TOPOLOGY* A pointer to the mesh topology to calculate the elements surrounding each node for

*ERR* The error code

*ERROR* The error string

Definition at line 6318 of file mesh\_routines.f90.

References *BASE\_ROUTINES::ENTERS()*, *BASE\_ROUTINES::ERRORS()*, and *BASE\_ROUTINES::EXITS()*.

Referenced by *MESH\_TOPOLOGY\_CALCULATE()*.

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.35.2.71 subroutine MESH\_ROUTINES::MESH\_USER\_NUMBER\_FIND**  
*(INTEGER(INTG),intent(in)* *USER\_NUMBER*, *TYPE(REGION\_TYPE)*,*pointer REGION*, *TYPE(MESH\_TYPE)*,*pointer MESH*, *INTEGER(INTG),intent(out)* *ERR*,  
*TYPE(VARYING\_STRING),intent(out)* *ERROR*, *\*) [private]*

Finds and returns in *MESH* a pointer to the mesh identified by *USER\_NUMBER* in the given *REGION*. If no mesh with that number exists *MESH* is left nullified.

**Parameters:**

*USER\_NUMBER* The user number of the mesh to find

*REGION* The region containing the mesh

*MESH* On return, a pointer to the mesh of the specified user number. In no mesh with the specified user number exists the pointer is returned NULL.

*ERR* The error code

*ERROR* The error string

Definition at line 6473 of file mesh\_routines.f90.

References *BASE\_ROUTINES::ENTERS()*, *BASE\_ROUTINES::ERRORS()*, *BASE\_ROUTINES::EXITS()*, and *KINDS::PTR*.

Referenced by *MESH\_CREATE\_START()*, *MESH\_NUMBER\_OF\_COMPONENTS\_SET\_NUMBER()*, and *MESH\_NUMBER\_OF\_ELEMENTS\_SET\_NUMBER()*.

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.35.2.72 subroutine MESH\_ROUTINES::MESSES\_FINALISE (TYPE(REGION\_-  
TYPE),pointer *REGION*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Finalises the meshes in the given region.

### **Todo**

pass in meshes

#### **Parameters:**

***REGION*** A pointer to the region to finalise the meshes for.

***ERR*** The error code

***ERROR*** The error string

Definition at line 6519 of file mesh\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), MESH\_DESTROY(), and KINDS::PTR.

Referenced by REGION\_ROUTINES::REGION\_DESTROY().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.35.2.73 subroutine MESH\_ROUTINES::MESSES\_INITIALISE (TYPE(REGION\_-  
TYPE),pointer *REGION*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Initialises the meshes for the given region.

### **Todo**

finalise on error

#### **Parameters:**

***REGION*** A pointer to the region to initialise the meshes for.

***ERR*** The error code

***ERROR*** The error string

Definition at line 6555 of file mesh\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Referenced    by    REGION\_ROUTINES::REGION\_CREATE\_START(),    and    REGION\_-ROUTINES::REGION\_SUB\_REGION\_CREATE\_START().

Here is the call graph for this function:

Here is the caller graph for this function:

## 6.36 NODE\_ROUTINES Namespace Reference

This module handles all node routines.

### Functions

- subroutine [NODE\\_CHECK\\_EXISTS](#) (USER\_NUMBER, REGION, NODE\_EXISTS, GLOBAL\_NUMBER, ERR, ERROR,\*)
- subroutine [NODE\\_DESTROY](#) (NODE, ERR, ERROR,\*)
- subroutine [NODE\\_INITIAL\\_POSITION\\_SET](#) (GLOBAL\_NUMBER, INITIAL\_POSITION, NODES, ERR, ERROR,\*)
- subroutine [NODE\\_NUMBER\\_SET](#) (GLOBAL\_NUMBER, USER\_NUMBER, NODES, ERR, ERROR,\*)
- subroutine [NODES\\_CREATE\\_FINISH](#) (REGION, ERR, ERROR,\*)
- subroutine [NODES\\_CREATE\\_START](#) (NUMBER\_OF\_NODES, REGION, NODES, ERR, ERROR,\*)
- subroutine [NODES\\_FINALISE](#) (REGION, ERR, ERROR,\*)
- subroutine [NODES\\_INITIALISE](#) (REGION, ERR, ERROR,\*)

### 6.36.1 Detailed Description

This module handles all node routines.

### 6.36.2 Function Documentation

**6.36.2.1 subroutine NODE\_ROUTINES::NODE\_CHECK\_EXISTS (INTEGER(INTG) *USER\_NUMBER*, TYPE(REGION\_TYPE),pointer *REGION*, LOGICAL,intent(out) *NODE\_EXISTS*, INTEGER(INTG),intent(out) *GLOBAL\_NUMBER*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Definition at line 75 of file node\_routines.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_ROUTINES::EXITS(), TREES::TREE\_NODE\_VALUE\_GET(), and TREES::TREE\_SEARCH().

Referenced      by      LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_FIXED\_CONDITIONS\_(), and MESH\_ROUTINES::MESH\_TOPOLOGY\_ELEMENTS\_ELEMENT\_NODES\_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.36.2.2 subroutine NODE\_ROUTINES::NODE\_DESTROY (TYPE(NODE\_TYPE) *NODE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Definition at line 126 of file node\_routines.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      and      BASE\_ROUTINES::EXITS().

Referenced by NODES\_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.36.2.3 subroutine NODE\_ROUTINES::NODE\_INITIAL\_POSITION\_SET**  
`(INTEGER(INTG),intent(in) GLOBAL_NUMBER, REAL(DP),dimension(:),intent(in)  
INITIAL_POSITION, TYPE(NODES_TYPE),pointer NODES,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)  
[private]`

Definition at line 153 of file node\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by FIELD\_ROUTINES::FIELD\_INTERPOLATION\_PARAMETERS\_LINE\_GET().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.36.2.4 subroutine NODE\_ROUTINES::NODE\_NUMBER\_SET** (INTEGER(INTG),intent(in)  
`GLOBAL_NUMBER, INTEGER(INTG),intent(in) USER_NUMBER,  
TYPE(NODES_TYPE),pointer NODES, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 216 of file node\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), TREES::TREE\_ITEM\_DELETE(), TREES::TREE\_ITEM\_INSERT(), and TREES::TREE\_NODE\_INSERT\_SUCESSFUL.

Referenced by FIELD\_IO\_ROUTINES::FIELD\_IO\_IMPORT\_GLOBAL\_MESH().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.36.2.5 subroutine NODE\_ROUTINES::NODES\_CREATE\_FINISH**  
`(TYPE(REGION_TYPE),pointer REGION, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 267 of file node\_routines.f90.

References BASE\_ROUTINES::DIAGNOSTIC\_OUTPUT\_TYPE, BASE\_ROUTINES::DIAGNOSTICS1, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and TREES::TREE\_OUTPUT().

Referenced by FIELD\_ROUTINES::FIELD\_INTERPOLATION\_PARAMETERS\_LINE\_GET(), and FIELD\_IO\_ROUTINES::FIELD\_IO\_IMPORT\_GLOBAL\_MESH().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.36.2.6 subroutine NODE\_ROUTINES::NODES\_CREATE\_START**  
**(INTEGER(INTG),intent(in) NUMBER\_OF\_NODES, TYPE(REGION\_TYPE),pointer REGION, TYPE(NODES\_TYPE),pointer NODES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Definition at line 322 of file node\_routines.f90.

References KINDS::\_DP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), TREES::TREE\_CREATE\_FINISH(), TREES::TREE\_CREATE\_START(), TREES::TREE\_INSERT\_TYPE\_SET(), TREES::TREE\_ITEM\_INSERT(), and TREES::TREE\_NO\_DUPLICATES\_ALLOWED.

Referenced by FIELD\_IO\_ROUTINES::FIELD\_IO\_IMPORT\_GLOBAL\_MESH(), and GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_REGULAR\_CREATE\_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.36.2.7 subroutine NODE\_ROUTINES::NODES\_FINALISE (TYPE(REGION\_TYPE),pointer REGION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Definition at line 404 of file node\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), NODE\_DESTROY(), and TREES::TREE\_DESTROY().

Referenced by REGION\_ROUTINES::REGION\_DESTROY().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.36.2.8 subroutine NODE\_ROUTINES::NODES\_INITIALISE (TYPE(REGION\_TYPE),pointer REGION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Definition at line 443 of file node\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by REGION\_ROUTINES::REGION\_CREATE\_START(), and REGION\_ROUTINES::REGION\_SUB\_REGION\_CREATE\_START().

Here is the call graph for this function:

Here is the caller graph for this function:

## 6.37 PROBLEM\_CONSTANTS Namespace Reference

This module handles all problem wide constants.

### Variables

- INTEGER(INTG), parameter `PROBLEM_NO_CLASS` = 0
- INTEGER(INTG), parameter `PROBLEM_ELASTICITY_CLASS` = 1
- INTEGER(INTG), parameter `PROBLEM_FLUID_MECHANICS_CLASS` = 2
- INTEGER(INTG), parameter `PROBLEM_ELECTROMAGNETICS_CLASS` = 3
- INTEGER(INTG), parameter `PROBLEM_CLASSICAL_FIELD_CLASS` = 4
- INTEGER(INTG), parameter `PROBLEM_MODAL_CLASS` = 5
- INTEGER(INTG), parameter `PROBLEM_FITTING_CLASS` = 6
- INTEGER(INTG), parameter `PROBLEM_OPTIMISATION_CLASS` = 7
- INTEGER(INTG), parameter `PROBLEM_NO_TYPE` = 0
- INTEGER(INTG), parameter `PROBLEM_LINEAR_ELASTICITY_TYPE` = 1
- INTEGER(INTG), parameter `PROBLEMFINITE_ELASTICITY_TYPE` = 2
- INTEGER(INTG), parameter `PROBLEM_STOKES_FLUID_TYPE` = 1
- INTEGER(INTG), parameter `PROBLEM_NAVIER_STOKES_FLUID_TYPE` = 2
- INTEGER(INTG), parameter `PROBLEM_ELECTROSTATIC_TYPE` = 1
- INTEGER(INTG), parameter `PROBLEM_MAGNETOSTATIC_TYPE` = 2
- INTEGER(INTG), parameter `PROBLEM_MAXWELLS_EQUATIONS_TYPE` = 3
- INTEGER(INTG), parameter `PROBLEM_LAPLACE_EQUATION_TYPE` = 1
- INTEGER(INTG), parameter `PROBLEM_POISSON_EQUATION_TYPE` = 2
- INTEGER(INTG), parameter `PROBLEM_HELMHOLTZ_EQUATION_TYPE` = 3
- INTEGER(INTG), parameter `PROBLEM_WAVE_EQUATION_TYPE` = 4
- INTEGER(INTG), parameter `PROBLEM_DIFFUSION_EQUATION_TYPE` = 5
- INTEGER(INTG), parameter `PROBLEM_ADVECTION_DIFFUSION_EQUATION_TYPE` = 6
- INTEGER(INTG), parameter `PROBLEMREACTION_DIFFUSION_EQUATION_TYPE` = 7
- INTEGER(INTG), parameter `PROBLEM_BIHAMONIC_EQUATION_TYPE` = 8
- INTEGER(INTG), parameter `PROBLEM_LINEAR_ELASTIC_MODAL_TYPE` = 1
- INTEGER(INTG), parameter `PROBLEM_NO_SUBTYPE` = 0
- INTEGER(INTG), parameter `PROBLEM_STANDARD_LAPLACE_SUBTYPE` = 1
- INTEGER(INTG), parameter `PROBLEM_GENERALISED_LAPLACE_SUBTYPE` = 2
- INTEGER(INTG), parameter `PROBLEM_SETUP_INITIAL_TYPE` = 1

*Initial setup for a problem.*

- INTEGER(INTG), parameter `PROBLEM_SETUP_CONTROL_TYPE` = 2

*Solver setup for a problem.*

- INTEGER(INTG), parameter `PROBLEM_SETUP SOLUTION_TYPE` = 3

*Solution parameters setup for a problem.*

- INTEGER(INTG), parameter `PROBLEM_SETUP_SOLVER_TYPE` = 4

*Solver setup for a problem.*

- INTEGER(INTG), parameter `PROBLEM_SETUP_START_ACTION` = 1

*Start setup action.*

- INTEGER(INTG), parameter **PROBLEM\_SETUP\_FINISH\_ACTION** = 2  
*Finish setup action.*
- INTEGER(INTG), parameter **PROBLEM\_SETUP\_DO\_ACTION** = 3  
*Do setup action.*
- INTEGER(INTG), parameter **PROBLEM\_SOLUTION\_NO\_OUTPUT** = 0  
*No output.*
- INTEGER(INTG), parameter **PROBLEM\_SOLUTION\_TIMING\_OUTPUT** = 1  
*Timing information output.*
- INTEGER(INTG), parameter **PROBLEM\_SOLUTION\_MATRIX\_OUTPUT** = 2  
*All below and solution matrices output.*
- INTEGER(INTG), parameter **PROBLEM\_SOLUTION\_LINEAR** = 1  
*The problem solution is linear.*
- INTEGER(INTG), parameter **PROBLEM\_SOLUTION\_NONLINEAR** = 2  
*The problem solution is nonlinear.*
- INTEGER(INTG), parameter **PROBLEM\_SOLVER\_NO\_OUTPUT** = 0  
*No output.*
- INTEGER(INTG), parameter **PROBLEM\_SOLVER\_TIMING\_OUTPUT** = 1  
*Timing information output.*
- INTEGER(INTG), parameter **PROBLEM\_SOLVER\_SOLVER\_OUTPUT** = 2  
*All below and solver output.*
- INTEGER(INTG), parameter **PROBLEM\_SOLVER\_SPARSE\_MATRICES** = 1  
*Use sparse solver matrices.*
- INTEGER(INTG), parameter **PROBLEM\_SOLVER\_FULL\_MATRICES** = 2  
*Use fully populated solver matrices.*

### 6.37.1 Detailed Description

This module handles all problem wide constants.

### 6.37.2 Variable Documentation

#### 6.37.2.1 INTEGER(INTG),parameter **PROBLEM\_CONSTANTS::PROBLEM\_ADVECTION\_DIFFUSION\_EQUATION\_TYPE** = 6

Definition at line 80 of file problem\_constants.f90.

Referenced by CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_PROBLEM\_CLASS\_TYPE\_SET(), and CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_PROBLEM\_SETUP().

**6.37.2.2 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_BIHARMONIC\_EQUATION\_TYPE = 8**

Definition at line 82 of file problem\_constants.f90.

Referenced by CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_PROBLEM\_CLASS\_TYPE\_SET(), and CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_PROBLEM\_SETUP().

**6.37.2.3 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_CLASSICAL\_FIELD\_CLASS = 4**

Definition at line 57 of file problem\_constants.f90.

Referenced by CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_PROBLEM\_CLASS\_TYPE\_GET(), LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_PROBLEM\_SUBTYPE\_SET(), PROBLEM\_ROUTINES::PROBLEM\_CREATE\_START(), PROBLEM\_ROUTINES::PROBLEM\_SETUP(), PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_GET\_PTR(), and PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_SET\_PTR().

**6.37.2.4 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_DIFFUSION\_EQUATION\_TYPE = 5**

Definition at line 79 of file problem\_constants.f90.

Referenced by CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_PROBLEM\_CLASS\_TYPE\_SET(), and CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_PROBLEM\_SETUP().

**6.37.2.5 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_ELASTICITY\_CLASS = 1**

Definition at line 54 of file problem\_constants.f90.

Referenced by FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITY\_PROBLEM\_SUBTYPE\_SET(), PROBLEM\_ROUTINES::PROBLEM\_SETUP(), PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_GET\_PTR(), and PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_SET\_PTR().

**6.37.2.6 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_ELECTROMAGNETICS\_CLASS = 3**

Definition at line 56 of file problem\_constants.f90.

Referenced by PROBLEM\_ROUTINES::PROBLEM\_SETUP(), PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_GET\_PTR(), and PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_SET\_PTR().

**6.37.2.7 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_ELECTROSTATIC\_TYPE = 1**

Definition at line 71 of file problem\_constants.f90.

**6.37.2.8 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEMFINITE\_-  
ELASTICITY\_TYPE = 2**

Definition at line 66 of file problem\_constants.f90.

Referenced by FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITY\_PROBLEM\_SUBTYPE\_SET().

**6.37.2.9 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEMFITTING\_-  
CLASS = 6**

Definition at line 59 of file problem\_constants.f90.

Referenced by PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_GET\_PTR(), and PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_SET\_PTR().

**6.37.2.10 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEMFLUID\_-  
MECHANICS\_CLASS = 2**

Definition at line 55 of file problem\_constants.f90.

Referenced by PROBLEM\_ROUTINES::PROBLEM\_SETUP(), PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_GET\_PTR(), and PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_SET\_PTR().

**6.37.2.11 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_-  
GENERALISED\_LAPLACE\_SUBTYPE = 2**

Definition at line 94 of file problem\_constants.f90.

Referenced by LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_PROBLEM\_SETUP(), and LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_PROBLEM\_SUBTYPE\_SET().

**6.37.2.12 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_-  
HELMHOLTZ\_EQUATION\_TYPE = 3**

Definition at line 77 of file problem\_constants.f90.

Referenced by CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_PROBLEM\_CLASS\_TYPE\_SET(), and CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_PROBLEM\_SETUP().

**6.37.2.13 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_LAPLACE\_-  
EQUATION\_TYPE = 1**

Definition at line 75 of file problem\_constants.f90.

Referenced by CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_PROBLEM\_CLASS\_TYPE\_SET(), CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_PROBLEM\_SETUP(), LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_PROBLEM\_SUBTYPE\_SET(), and PROBLEM\_ROUTINES::PROBLEM\_CREATE\_START().

**6.37.2.14 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_LINEAR\_-  
ELASTIC\_MODAL\_TYPE = 1**

Definition at line 84 of file problem\_constants.f90.

**6.37.2.15 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_LINEAR\_-  
ELASTICITY\_TYPE = 1**

Definition at line 65 of file problem\_constants.f90.

**6.37.2.16 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_-  
MAGNETOSTATIC\_TYPE = 2**

Definition at line 72 of file problem\_constants.f90.

**6.37.2.17 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_MAXWELLS\_-  
EQUATIONS\_TYPE = 3**

Definition at line 73 of file problem\_constants.f90.

**6.37.2.18 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_MODAL\_-  
CLASS = 5**

Definition at line 58 of file problem\_constants.f90.

Referenced by PROBLEM\_ROUTINES::PROBLEM\_SETUP(), PROBLEM\_ROUTINES::PROBLEM\_-  
SPECIFICATION\_GET\_PTR(), and PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_SET\_-  
PTR().

**6.37.2.19 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_NAVIER\_-  
STOKES\_FLUID\_TYPE = 2**

Definition at line 69 of file problem\_constants.f90.

**6.37.2.20 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_NO\_CLASS = 0**

Definition at line 53 of file problem\_constants.f90.

Referenced by PROBLEM\_ROUTINES::PROBLEM\_INITIALISE().

**6.37.2.21 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_NO\_SUBTYPE  
= 0**

Definition at line 87 of file problem\_constants.f90.

Referenced by FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITY\_PROBLEM\_SETUP(),  
FINITE\_ELASTICITY\_ROUTINES::FINITE\_ELASTICITY\_PROBLEM\_SUBTYPE\_SET(),  
LINEAR\_ELASTICITY\_ROUTINES::LINEAR\_ELASTICITY\_PROBLEM\_SUBTYPE\_SET(), and  
PROBLEM\_ROUTINES::PROBLEM\_INITIALISE().

**6.37.2.22 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_NO\_TYPE = 0**

Definition at line 63 of file problem\_constants.f90.

Referenced by PROBLEM\_ROUTINES::PROBLEM\_INITIALISE().

**6.37.2.23 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_OPTIMISATION\_CLASS = 7**

Definition at line 60 of file problem\_constants.f90.

Referenced by PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_GET\_PTR(), and PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_SET\_PTR().

**6.37.2.24 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_POISSON\_EQUATION\_TYPE = 2**

Definition at line 76 of file problem\_constants.f90.

Referenced by CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_PROBLEM\_CLASS\_TYPE\_SET(), and CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_PROBLEM\_SETUP().

**6.37.2.25 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEMREACTION\_DIFFUSION\_EQUATION\_TYPE = 7**

Definition at line 81 of file problem\_constants.f90.

Referenced by CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_PROBLEM\_CLASS\_TYPE\_SET(), and CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_PROBLEM\_SETUP().

**6.37.2.26 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_STANDARD\_LAPLACE\_SUBTYPE = 1**

Definition at line 93 of file problem\_constants.f90.

Referenced by LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_PROBLEM\_SETUP(), LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_STANDARD\_SETUP(), LAPLACE\_EQUATIONS\_ROUTINES::LAPLACE\_EQUATION\_PROBLEM\_SUBTYPE\_SET(), and PROBLEM\_ROUTINES::PROBLEM\_CREATE\_START().

**6.37.2.27 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_STOKES\_FLUID\_TYPE = 1**

Definition at line 68 of file problem\_constants.f90.

**6.37.2.28 INTEGER(INTG),parameter PROBLEM\_CONSTANTS::PROBLEM\_WAVE\_EQUATION\_TYPE = 4**

Definition at line 78 of file problem\_constants.f90.

Referenced by CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_PROBLEM\_CLASS\_TYPE\_SET(), and CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_PROBLEM\_SETUP().

## 6.38 PROBLEM\_ROUTINES Namespace Reference

This module handles all problem routines.

### Classes

- interface [PROBLEM\\_DESTROY](#)
- interface [PROBLEM\\_SPECIFICATION\\_GET](#)
- interface [PROBLEM\\_SPECIFICATION\\_SET](#)

### Functions

- subroutine [PROBLEM\\_CREATE\\_FINISH](#) (PROBLEM, ERR, ERROR,\*)
 

*Finishes the process of creating a problem.*
- subroutine [PROBLEM\\_CREATE\\_START](#) (USER\_NUMBER, PROBLEM, ERR, ERROR,\*)
 

*Starts the process of creating a problem defined by USER\_NUMBER.*
- subroutine [PROBLEM\\_DESTROY\\_NUMBER](#) (USER\_NUMBER, ERR, ERROR,\*)
 

*Destroys a problem identified by a user number.*
- subroutine [PROBLEM\\_DESTROY\\_PTR](#) (PROBLEM, ERR, ERROR,\*)
 

*Destroys a problem identified by a pointer.*
- subroutine [PROBLEM\\_FINALISE](#) (PROBLEM, ERR, ERROR,\*)
 

*Finalise the problem and deallocate all memory.*
- subroutine [PROBLEM\\_INITIALISE](#) (PROBLEM, ERR, ERROR,\*)
 

*Initialises a problem.*
- subroutine [PROBLEM\\_CONTROL\\_CREATE\\_FINISH](#) (PROBLEM, ERR, ERROR,\*)
 

*Finish the creation of the control for the problem.*
- subroutine [PROBLEM\\_CONTROL\\_CREATE\\_START](#) (PROBLEM, ERR, ERROR,\*)
 

*Start the creation of a problem control for a problem.*
- subroutine [PROBLEM\\_CONTROL\\_DESTROY](#) (PROBLEM, ERR, ERROR,\*)
 

*Destroy the control for a problem.*
- subroutine [PROBLEM\\_CONTROL\\_FINALISE](#) (CONTROL, ERR, ERROR,\*)
 

*Finalise the control for a problem and deallocate all memory.*
- subroutine [PROBLEM\\_CONTROL\\_INITIALISE](#) (PROBLEM, ERR, ERROR,\*)
 

*Initialise the control for a problem.*
- subroutine [PROBLEM\\_SOLUTION\\_JACOBIAN\\_EVALUATE](#) (SOLUTION, ERR, ERROR,\*)
 

*Evaluates the Jacobian for a nonlinear problem solution.*
- subroutine [PROBLEM\\_SOLUTION\\_RESIDUAL\\_EVALUATE](#) (SOLUTION, ERR, ERROR,\*)
 

*Evaluates the residual for a nonlinear problem solution.*

*Evaluates the residual for a nonlinear problem solution.*

- subroutine **PROBLEM\_SETUP** (PROBLEM, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)  
*Sets up the specifics for a problem.*
- subroutine **PROBLEM\_SOLVE** (PROBLEM, ERR, ERROR,\*)  
*Solves a problem.*
- subroutine **PROBLEM SOLUTION\_SOLVE** (SOLUTION, ERR, ERROR,\*)  
*Solves a solution.*
- subroutine **PROBLEM SOLUTIONS\_CREATE\_FINISH** (PROBLEM, ERR, ERROR,\*)  
*Finish the creation of solutions for a problem.*
- subroutine **PROBLEM SOLUTIONS\_CREATE\_START** (PROBLEM, ERR, ERROR,\*)  
*Start the creation of a solution for the problem.*
- subroutine **PROBLEM SOLUTION\_EQUATIONS\_SET\_ADD** (PROBLEM, SOLUTION\_INDEX, EQUATIONS\_SET, EQUATIONS\_SET\_INDEX, ERR, ERROR,\*)  
*Adds an equations set to a problem solution.*
- subroutine **PROBLEM SOLUTION\_FINALISE** (SOLUTION, ERR, ERROR,\*)  
*Finalises a solution and deallocates all memory.*
- subroutine **PROBLEM SOLUTION\_INITIALISE** (SOLUTION, ERR, ERROR,\*)  
*Initialises the solution for a problem.*
- subroutine **PROBLEM SOLUTIONS\_FINALISE** (PROBLEM, ERR, ERROR,\*)  
*Finalises the solutions for a problem and deallocates all memory.*
- subroutine **PROBLEM SOLUTIONS\_INITIALISE** (PROBLEM, ERR, ERROR,\*)  
*Initialises the solutions for a problem.*
- subroutine **PROBLEM SOLVER\_CREATE\_FINISH** (PROBLEM, ERR, ERROR,\*)  
*Finish the creation of the solver for the problem solutions.*
- subroutine **PROBLEM SOLVER\_CREATE\_START** (PROBLEM, ERR, ERROR,\*)  
*Start the creation of a problem solver for a problem.*
- subroutine **PROBLEM SOLVER\_DESTROY** (PROBLEM, ERR, ERROR,\*)  
*Destroy the solvers for a problem.*
- subroutine **PROBLEM SOLVER\_GET** (PROBLEM, SOLUTION\_INDEX, SOLVER, ERR, ERROR,\*)  
*Returns a pointer to the solver for a solution on a problem.*
- subroutine **PROBLEM SPECIFICATION\_GET\_NUMBER** (USER\_NUMBER, PROBLEM\_CLASS, PROBLEM\_EQUATION\_TYPE, PROBLEM\_SUBTYPE, ERR, ERROR,\*)  
*Gets the problem specification i.e., problem class, type and subtype for a problem identified by a user number.*

- subroutine **PROBLEM\_SPECIFICATION\_GET\_PTR** (PROBLEM, PROBLEM\_CLASS, PROBLEM\_EQUATION\_TYPE, PROBLEM\_SUBTYPE, ERR, ERROR,\*)
 

*Gets the problem specification i.e., problem class, type and subtype for a problem identified by a pointer.*
- subroutine **PROBLEM\_SPECIFICATION\_SET\_NUMBER** (USER\_NUMBER, PROBLEM\_CLASS, PROBLEM\_EQUATION\_TYPE, PROBLEM\_SUBTYPE, ERR, ERROR,\*)
 

*Sets/changes the problem specification i.e., problem class, type and subtype for a problem identified by a user number.*
- subroutine **PROBLEM\_SPECIFICATION\_SET\_PTR** (PROBLEM, PROBLEM\_CLASS, PROBLEM\_EQUATION\_TYPE, PROBLEM\_SUBTYPE, ERR, ERROR,\*)
 

*Sets/changes the problem specification i.e., problem class, type and subtype for a problem identified by a pointer.*
- subroutine **PROBLEM\_USER\_NUMBER\_FIND** (USER\_NUMBER, PROBLEM, ERR, ERROR,\*)
 

*Finds and returns in PROBLEM a pointer to the problem identified by USER\_NUMBER. If no problem with that USER\_NUMBER exists PROBLEM is left nullified.*
- subroutine **PROBLEMS\_FINALISE** (ERR, ERROR,\*)
 

*Finalises all problems and deallocates all memory.*
- subroutine **PROBLEMS\_INITIALISE** (ERR, ERROR,\*)
 

*Initialises all problems.*

## Variables

- INTEGER(INTG), parameter **NUMBER\_OF\_PROBLEM\_LINEARITIES** = 3
 

*The number of problem linearity types defined.*
- INTEGER(INTG), parameter **PROBLEM\_LINEAR** = 1
 

*The problem is linear.*
- INTEGER(INTG), parameter **PROBLEM\_NONLINEAR** = 2
 

*The problem is non-linear.*
- INTEGER(INTG), parameter **PROBLEM\_NONLINEAR\_BCS** = 3
 

*The problem has non-linear boundary conditions.*
- INTEGER(INTG), parameter **NUMBER\_OF\_PROBLEM\_TIME\_TYPES** = 3
 

*The number of problem time dependence types defined.*
- INTEGER(INTG), parameter **PROBLEM\_STATIC** = 1
 

*The problem is static and has no time dependence.*
- INTEGER(INTG), parameter **PROBLEM\_DYNAMIC** = 2
 

*The problem is dynamic.*

- INTEGER(INTG), parameter **PROBLEM\_QUASISTATIC** = 3  
*The problem is quasi-static.*
- TYPE(**PROBLEMS\_TYPE**), target **PROBLEMS**

### 6.38.1 Detailed Description

This module handles all problem routines.

### 6.38.2 Function Documentation

#### 6.38.2.1 subroutine PROBLEM\_ROUTINES::PROBLEM\_CONTROL\_CREATE\_FINISH (TYPE(PROBLEM\_TYPE),pointer *PROBLEM*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Finish the creation of the control for the problem.

**Parameters:**

- PROBLEM** A pointer to the problem to finish the control for  
**ERR** The error code  
**ERROR** The error string

Definition at line 437 of file problem\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(),    PROBLEM\_SETUP(),    PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_-CONTROL\_TYPE, and PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_FINISH\_ACTION.

Here is the call graph for this function:

#### 6.38.2.2 subroutine PROBLEM\_ROUTINES::PROBLEM\_CONTROL\_CREATE\_START (TYPE(PROBLEM\_TYPE),pointer *PROBLEM*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Start the creation of a problem control for a problem.

**Parameters:**

- PROBLEM** A pointer to the problem to start the creation of a control for.  
**ERR** The error code  
**ERROR** The error string

Definition at line 476 of file problem\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), PROBLEM\_CONTROL\_FINALISE(), PROBLEM\_CONTROL\_INITIALISE(), PROBLEM\_SETUP(), PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_CONTROL\_TYPE, and PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_START\_ACTION.

Here is the call graph for this function:

**6.38.2.3 subroutine PROBLEM\_ROUTINES::PROBLEM\_CONTROL\_DESTROY**  
 (TYPE(PROBLEM\_TYPE),pointer *PROBLEM*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Destroy the control for a problem.

**Parameters:**

- PROBLEM*** A pointer to the problem to destroy the control for.
- ERR*** The error code
- ERROR*** The error string

Definition at line 514 of file problem\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), and PROBLEM\_CONTROL\_FINALISE().

Here is the call graph for this function:

**6.38.2.4 subroutine PROBLEM\_ROUTINES::PROBLEM\_CONTROL\_FINALISE**  
 (TYPE(PROBLEM\_CONTROL\_TYPE),pointer *CONTROL*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)  
 [private]

Finalise the control for a problem and deallocate all memory.

**Parameters:**

- CONTROL*** A pointer to the problem control to finalise.
- ERR*** The error code
- ERROR*** The error string

Definition at line 546 of file problem\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Referenced by PROBLEM\_CONTROL\_CREATE\_START(), PROBLEM\_CONTROL\_DESTROY(), PROBLEM\_CONTROL\_INITIALISE(), and PROBLEM\_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.38.2.5 subroutine PROBLEM\_ROUTINES::PROBLEM\_CONTROL\_INITIALISE**  
 (TYPE(PROBLEM\_TYPE),pointer *PROBLEM*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Initialise the control for a problem.

**Parameters:**

- PROBLEM*** A pointer to the problem to initialise the control for.
- ERR*** The error code

**ERROR** The error string

Definition at line 571 of file problem\_routines.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_-ROUTINES::EXITS(), and PROBLEM\_CONTROL\_FINALISE().

Referenced by PROBLEM\_CONTROL\_CREATE\_START().

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.38.2.6 subroutine PROBLEM\_ROUTINES::PROBLEM\_CREATE\_FINISH (**TYPE(PROBLEM\_TYPE)**,pointer **PROBLEM**, **INTEGER(INTG)**,intent(out) **ERR**, **TYPE(VARYING\_STRING)**,intent(out) **ERROR**, \*)

Finishes the process of creating a problem.

##### Parameters:

**PROBLEM** A pointer to the problem to finish creating.

**ERR** The error code

**ERROR** The error string

Definition at line 140 of file problem\_routines.f90.

References      BASE\_ROUTINES::DIAGNOSTIC\_OUTPUT\_TYPE,      BASE\_-ROUTINES::DIAGNOSTICS1,      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),  
BASE\_ROUTINES::EXITS(), PROBLEM\_SETUP(), PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_-FINISH\_ACTION, PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_INITIAL\_TYPE, PROBLEM\_-SOLUTIONS\_INITIALISE(), PROBLEMS, and KINDS::PTR.

Here is the call graph for this function:

#### 6.38.2.7 subroutine PROBLEM\_ROUTINES::PROBLEM\_CREATE\_START (**INTEGER(INTG)**,intent(in) **USER\_NUMBER**, **TYPE(PROBLEM\_TYPE)**,pointer **PROBLEM**, **INTEGER(INTG)**,intent(out) **ERR**, **TYPE(VARYING\_STRING)**,intent(out) **ERROR**, \*)

Starts the process of creating a problem defined by **USER\_NUMBER**.

##### Parameters:

**USER\_NUMBER** The user number of the problem to create

**PROBLEM** On return, a pointer to the created problem. Must not be associated on entry.

**ERR** The error code

**ERROR** The error string

Definition at line 192 of file problem\_routines.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_-ROUTINES::EXITS(),      PROBLEM\_CONSTANTS::PROBLEM\_CLASSICAL\_FIELD\_CLASS,  
PROBLEM\_INITIALISE(), PROBLEM\_CONSTANTS::PROBLEM\_LAPLACE\_EQUATION\_TYPE,

PROBLEM\_SETUP(), PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_INITIAL\_TYPE, PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_START\_ACTION, PROBLEM\_CONSTANTS::PROBLEM\_STANDARD\_LAPLACE\_SUBTYPE, PROBLEM\_USER\_NUMBER\_FIND(), PROBLEMS, and KINDS::PTR.

Here is the call graph for this function:

**6.38.2.8 subroutine PROBLEM\_ROUTINES::PROBLEM\_DESTROY\_NUMBER**  
**(INTEGER(INTG),intent(in) *USER\_NUMBER*, INTEGER(INTG),intent(out) *ERR*,**  
**TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Destroys a problem identified by a user number.

**Parameters:**

***USER\_NUMBER*** The user number of the problem to destroy  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 261 of file problem\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), PROBLEMS, and KINDS::PTR.

Here is the call graph for this function:

**6.38.2.9 subroutine PROBLEM\_ROUTINES::PROBLEM\_DESTROY\_PTR**  
**(TYPE(PROBLEM\_TYPE),pointer *PROBLEM*, INTEGER(INTG),intent(out) *ERR*,**  
**TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Destroys a problem identified by a pointer.

**Parameters:**

***PROBLEM*** A pointer to the problem to destroy  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 312 of file problem\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), PROBLEM\_FINALISE(), PROBLEMS, and KINDS::PTR.

Here is the call graph for this function:

**6.38.2.10 subroutine PROBLEM\_ROUTINES::PROBLEM\_FINALISE**  
**(TYPE(PROBLEM\_TYPE),pointer *PROBLEM*, INTEGER(INTG),intent(out) *ERR*,**  
**TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Finalise the problem and deallocate all memory.

**Parameters:**

***PROBLEM*** A pointer to the problem to finalise.

***ERR*** The error code

***ERROR*** The error string

Definition at line 374 of file problem\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `PROBLEM_CONTROL_FINALISE()`,    and    `PROBLEM_SOLUTIONS_FINALISE()`.

Referenced by `PROBLEM_DESTROY_PTR()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.38.2.11 subroutine `PROBLEM_ROUTINES::PROBLEM_INITIALISE`

```
(TYPE(PROBLEM_TYPE),pointer PROBLEM, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Initialises a problem.

##### Parameters:

***PROBLEM*** The pointer to the problem

***ERR*** The error code !<The error code

***ERROR*** The error string !<The error string

Definition at line 402 of file problem\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `PROBLEM_CONSTANTS::PROBLEM_NO_CLASS`,    `PROBLEM_CONSTANTS::PROBLEM_NO_SUBTYPE`,    `PROBLEM_CONSTANTS::PROBLEM_NO_TYPE`, and `PROBLEMS`.

Referenced by `PROBLEM_CREATE_START()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.38.2.12 subroutine `PROBLEM_ROUTINES::PROBLEM_SETUP` (**TYPE(PROBLEM\_TYPE),pointer** *PROBLEM*, **INTEGER(INTG),intent(in)** *SETUP\_TYPE*,

```
INTEGER(INTG),intent(in) ACTION_TYPE, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Sets up the specifics for a problem.

##### Parameters:

***PROBLEM*** A pointer to the problem to setup

***SETUP\_TYPE*** The problem setup type

**See also:**

[PROBLEM\\_CONSTANTS::SetupTypes](#),[PROBLEM\\_CONSTANTS](#)

***ACTION\_TYPE*** The problem setup action type

See also:

[PROBLEM\\_CONSTANTS::SetupActionTypes](#), [CONSTANTS\\_ROUTINES](#)

**ERR** The error code

**ERROR** The error string

Definition at line 735 of file problem\_routines.f90.

References CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_PROBLEM\_SETUP(), ELASTICITY\_ROUTINES::ELASTICITY\_PROBLEM\_SETUP(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), PROBLEM\_CONSTANTS::PROBLEM\_CLASSICAL\_FIELD\_CLASS, PROBLEM\_CONSTANTS::PROBLEM\_ELASTICITY\_CLASS, PROBLEM\_CONSTANTS::PROBLEM\_ELECTROMAGNETICS\_CLASS, PROBLEM\_CONSTANTS::PROBLEM\_FLUID\_MECHANICS\_CLASS, and PROBLEM\_CONSTANTS::PROBLEM\_MODAL\_CLASS.

Referenced by PROBLEM\_CONTROL\_CREATE\_FINISH(), PROBLEM\_CONTROL\_CREATE\_START(), PROBLEM\_CREATE\_FINISH(), PROBLEM\_CREATE\_START(), PROBLEM\_SOLUTION\_EQUATIONS\_SET\_ADD(), PROBLEM\_SOLUTIONS\_CREATE\_FINISH(), PROBLEM\_SOLUTIONS\_CREATE\_START(), PROBLEM\_SOLVER\_CREATE\_FINISH(), and PROBLEM\_SOLVER\_CREATE\_START().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.38.2.13 subroutine PROBLEM\_ROUTINES::PROBLEM\_SOLUTION\_EQUATIONS\_SET\_ADD** (TYPE(PROBLEM\_TYPE),pointer **PROBLEM**, INTEGER(INTG),intent(in) **SOLUTION\_INDEX**, TYPE(EQUATIONS\_SET\_TYPE),pointer **EQUATIONS\_SET**, INTEGER(INTG),intent(out) **EQUATIONS\_SET\_INDEX**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

Adds an equations set to a problem solution.

#### Parameters:

**PROBLEM** A pointer to the problem to add the equations set to a solution

**SOLUTION\_INDEX** The solution index in the problem to add the equations set to

**EQUATIONS\_SET** A pointer to the equations set to add

**EQUATIONS\_SET\_INDEX** On return, the index of the equations set that has been added.

**ERR** The error code

**ERROR** The error string

Definition at line 970 of file problem\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), PROBLEM\_SETUP(), PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_DO\_ACTION, PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_SOLUTION\_TYPE, and KINDS::PTR.

Here is the call graph for this function:

**6.38.2.14 subroutine PROBLEM\_ROUTINES::PROBLEM\_SOLUTION\_FINALISE**  
`(TYPE(SOLUTION_TYPE),pointer SOLUTION, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Finalises a solution and deallocates all memory.

**Parameters:**

- SOLUTION*** A pointer to the solution to finalise
- ERR*** The error code
- ERROR*** The error string

Definition at line 1035 of file problem\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `SOLVER_ROUTINES::SOLVER_DESTROY()`.

Referenced by `PROBLEM_SOLUTIONS_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.38.2.15 subroutine PROBLEM\_ROUTINES::PROBLEM\_SOLUTION\_INITIALISE**  
`(TYPE(SOLUTION_TYPE),pointer SOLUTION, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Initialises the solution for a problem.

**Parameters:**

- SOLUTION*** A pointer to the solution to initialise
- ERR*** The error code
- ERROR*** The error string

Definition at line 1063 of file problem\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `PROBLEM_CONSTANTS::PROBLEM SOLUTION_LINEAR`.

Referenced by `PROBLEM_SOLUTIONS_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.38.2.16 subroutine PROBLEM\_ROUTINES::PROBLEM\_SOLUTION\_JACOBIAN\_EVALUATE**  
`(TYPE(SOLUTION_TYPE),pointer SOLUTION,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`  
`*)`

Evaluates the Jacobian for a nonlinear problem solution.

**Parameters:**

- SOLUTION*** A pointer to the solution to evaluate the Jacobian for

**ERR** The error code

**ERROR** The error string

Definition at line 609 of file problem\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `PROBLEM_CONSTANTS::PROBLEM SOLUTION_NONLINEAR`,    `KINDS::PTR`,    `SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE()`,    and    `SOLVER_ROUTINES::SOLVER_VARIABLES_UPDATE()`.

Referenced by `PROBLEM SOLUTION_JACOBIAN_EVALUATE_PETSC()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.38.2.17 subroutine PROBLEM\_ROUTINES::PROBLEM SOLUTION - RESIDUAL\_EVALUATE (TYPE(SOLUTION\_TYPE),pointer *SOLUTION*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Evaluates the residual for a nonlinear problem solution.

#### Parameters:

***SOLUTION*** A pointer to the solution to evaluate the residual for

***ERR*** The error code

***ERROR*** The error string

Definition at line 672 of file problem\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `PROBLEM_CONSTANTS::PROBLEM SOLUTION_NONLINEAR`,    `KINDS::PTR`,    `SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE()`,    and    `SOLVER_ROUTINES::SOLVER_VARIABLES_UPDATE()`.

Referenced by `PROBLEM SOLUTION_RESIDUAL_EVALUATE_PETSC()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.38.2.18 subroutine PROBLEM\_ROUTINES::PROBLEM SOLUTION\_SOLVE (TYPE(SOLUTION\_TYPE),pointer *SOLUTION*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Solves a solution.

#### Parameters:

***SOLUTION*** A pointer to the solution to solve

***ERR*** The error code

***ERROR*** The error string

Definition at line 828 of file problem\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `PROBLEM_CONSTANTS::PROBLEM SOLUTION_LINEAR`,    `KINDS::PTR`,    `SOLVER_ROUTINES::SOLVER_SOLVE()`,    and    `SOLVER_ROUTINES::SOLVER_VARIABLES_UPDATE()`.

Referenced by `PROBLEM_SOLVE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### **6.38.2.19 subroutine `PROBLEM_ROUTINES::PROBLEM SOLUTIONS CREATE FINISH`**

(`TYPE(PROBLEM_TYPE),pointer PROBLEM`, `INTEGER(INTG),intent(out) ERR`,  
`TYPE(VARYING_STRING),intent(out) ERROR, *`)

Finish the creation of solutions for a problem.

**Parameters:**

***PROBLEM*** A pointer to the problem

***ERR*** The error code

***ERROR*** The error string

Definition at line 893 of file `problem_routines.f90`.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `PROBLEM_SETUP()`,    `PROBLEM_CONSTANTS::PROBLEM SETUP FINISH ACTION`,    `PROBLEM_CONSTANTS::PROBLEM SETUP SOLUTION TYPE`,    and    `KINDS::PTR`.

Here is the call graph for this function:

#### **6.38.2.20 subroutine `PROBLEM_ROUTINES::PROBLEM SOLUTIONS CREATE START`**

(`TYPE(PROBLEM_TYPE),pointer PROBLEM`, `INTEGER(INTG),intent(out) ERR`,  
`TYPE(VARYING_STRING),intent(out) ERROR, *`)

Start the creation of a solution for the problem.

#### **Todo**

Should this return a pointer to the problem solution???

**Parameters:**

***PROBLEM*** A pointer to the problem to create a solution for

***ERR*** The error code !<The error code

***ERROR*** The error string !<The error string

Definition at line 937 of file `problem_routines.f90`.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `PROBLEM_SETUP()`,    `PROBLEM_CONSTANTS::PROBLEM SETUP SOLUTION TYPE`, and `PROBLEM_CONSTANTS::PROBLEM SETUP START ACTION`.

Here is the call graph for this function:

---

**6.38.2.21 subroutine PROBLEM\_ROUTINES::PROBLEM\_SOLUTIONS\_FINALISE**  
**(TYPE(PROBLEM\_TYPE),pointer *PROBLEM*, INTEGER(INTG),intent(out) *ERR*,**  
**TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Finalises the solutions for a problem and deallocates all memory.

**Parameters:**

***PROBLEM*** A pointer to the problem to finalise the solutions for

***ERR*** The error code

***ERROR*** The error string

Definition at line 1098 of file problem\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    **BASE\_-ROUTINES::EXITS()**, **PROBLEM\_SOLUTION\_FINALISE()**, and **KINDS::PTR**.

Referenced by **PROBLEM\_FINALISE()**, and **PROBLEM\_SOLUTIONS\_INITIALISE()**.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.38.2.22 subroutine PROBLEM\_ROUTINES::PROBLEM\_SOLUTIONS\_INITIALISE**  
**(TYPE(PROBLEM\_TYPE),pointer *PROBLEM*, INTEGER(INTG),intent(out) *ERR*,**  
**TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Initialises the solutions for a problem.

**Parameters:**

***PROBLEM*** A pointer to the problem to initialise the solutions for

***ERR*** The error code

***ERROR*** The error string

Definition at line 1133 of file problem\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    **BASE\_-ROUTINES::EXITS()**,    **PROBLEM\_SOLUTION\_INITIALISE()**,    **PROBLEM\_SOLUTIONS\_FINALISE()**, and **KINDS::PTR**.

Referenced by **PROBLEM\_CREATE\_FINISH()**.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.38.2.23 subroutine PROBLEM\_ROUTINES::PROBLEM\_SOLVE** (**TYPE(PROBLEM\_-TYPE),pointer *PROBLEM*, INTEGER(INTG),intent(out) *ERR*,**  
**TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Solves a problem.

**Parameters:**

***PROBLEM*** A pointer to the problem to solve.

**ERR** The error code

**ERROR** The error string

Definition at line 780 of file problem\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_-ROUTINES::EXITS()`, `PROBLEM_SOLUTION_SOLVE()`, and `KINDS::PTR`.

Here is the call graph for this function:

#### 6.38.2.24 subroutine `PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_FINISH`

```
(TYPE(PROBLEM_TYPE),pointer PROBLEM, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)
```

Finish the creation of the solver for the problem solutions.

**Parameters:**

**PROBLEM** A pointer to the problem to finish the solvers for

**ERR** The error code

**ERROR** The error string

Definition at line 1184 of file problem\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_-ROUTINES::EXITS()`,    `PROBLEM_SETUP()`,    `PROBLEM_CONSTANTS::PROBLEM_SETUP_FINISH_ACTION`, and `PROBLEM_CONSTANTS::PROBLEM_SETUP_SOLVER_TYPE`.

Here is the call graph for this function:

#### 6.38.2.25 subroutine `PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_START`

```
(TYPE(PROBLEM_TYPE),pointer PROBLEM, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)
```

Start the creation of a problem solver for a problem.

**Parameters:**

**PROBLEM** A pointer to the problem to start the creation of a solution for.

**ERR** The error code

**ERROR** The error string

Definition at line 1213 of file problem\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_-ROUTINES::EXITS()`,    `PROBLEM_SETUP()`,    `PROBLEM_CONSTANTS::PROBLEM_SETUP_SOLVER_TYPE`,    `PROBLEM_CONSTANTS::PROBLEM_SETUP_START_ACTION`,    and `KINDS::PTR`.

Here is the call graph for this function:

---

**6.38.2.26 subroutine PROBLEM\_ROUTINES::PROBLEM\_SOLVER\_DESTROY**  
 (*TYPE(PROBLEM\_TYPE),pointer PROBLEM, INTEGER(INTG),intent(out) ERR,*  
*TYPE(VARYING\_STRING),intent(out) ERROR, \*/*)

Destroy the solvers for a problem.

**Parameters:**

**PROBLEM** A pointer to the problem to destroy the solver for.

**ERR** The error code

**ERROR** The error string

Definition at line 1257 of file problem\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    **BASE\_-ROUTINES::EXITS()**, **KINDS::PTR**, and **SOLVER\_ROUTINES::SOLVER\_DESTROY()**.

Here is the call graph for this function:

**6.38.2.27 subroutine PROBLEM\_ROUTINES::PROBLEM\_SOLVER\_GET**  
 (*TYPE(PROBLEM\_TYPE),pointer PROBLEM, INTEGER(INTG),intent(in)*  
*SOLUTION\_INDEX, TYPE(SOLVER\_TYPE),pointer SOLVER,*  
*INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,*  
*\*)*)

Returns a pointer to the solver for a solution on a problem.

**Parameters:**

**PROBLEM** A pointer to the problem to get the solver for.

**SOLUTION\_INDEX** The solution index to get the solver for.

**SOLVER** On return, a pointer to the solver. Must not be associated on entry.

**ERR** The error code

**ERROR** The error string

Definition at line 1298 of file problem\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    **BASE\_-ROUTINES::EXITS()**, and **KINDS::PTR**.

Here is the call graph for this function:

**6.38.2.28 subroutine PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_GET\_NUMBER**  
 (*INTEGER(INTG),intent(in) USER\_NUMBER, INTEGER(INTG),intent(out)*  
*PROBLEM\_CLASS, INTEGER(INTG),intent(out) PROBLEM\_EQUATION\_TYPE,*  
*INTEGER(INTG),intent(out) PROBLEM\_SUBTYPE, INTEGER(INTG),intent(out)*  
*ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*/* [private])

Gets the problem specification i.e., problem class, type and subtype for a problem identified by a user number.

**Parameters:**

**USER\_NUMBER** The user number of the problem to set the specification for.

**PROBLEM\_CLASS** The problem class to get.

**PROBLEM\_EQUATION\_TYPE** The problem equation to get.

**PROBLEM\_SUBTYPE** The problem subtype.

**ERR** The error code

**ERROR** The error string

Definition at line 1346 of file problem\_routines.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_-ROUTINES::EXITS(), and PROBLEM\_USER\_NUMBER\_FIND().

Here is the call graph for this function:

```
6.38.2.29 subroutine PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_GET_PTR
 (TYPE(PROBLEM_TYPE),pointer PROBLEM, INTEGER(INTG),intent(out)
 PROBLEM_CLASS, INTEGER(INTG),intent(out) PROBLEM_EQUATION_TYPE,
 INTEGER(INTG),intent(out) PROBLEM_SUBTYPE, INTEGER(INTG),intent(out)
 ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Gets the problem specification i.e., problem class, type and subtype for a problem identified by a pointer.

#### Parameters:

**PROBLEM** A pointer to the problem to set the specification for.

**PROBLEM\_CLASS** The problem class to set.

**PROBLEM\_EQUATION\_TYPE** The problem equation type to set.

**PROBLEM\_SUBTYPE** The problem subtype to set.

**ERR** The error code

**ERROR** The error string

Definition at line 1375 of file problem\_routines.f90.

References     CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_PROBLEM\_CLASS\_TYPE\_-GET(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), PROBLEM\_CONSTANTS::PROBLEM\_CLASSICAL\_FIELD\_CLASS,     PROBLEM\_-CONSTANTS::PROBLEM\_ELASTICITY\_CLASS,     PROBLEM\_CONSTANTS::PROBLEM\_-ELECTROMAGNETICS\_CLASS,     PROBLEM\_CONSTANTS::PROBLEM\_FITTING\_-CLASS,     PROBLEM\_CONSTANTS::PROBLEM\_FLUID\_MECHANICS\_CLASS,     PROBLEM\_-CONSTANTS::PROBLEM\_MODAL\_CLASS,     and     PROBLEM\_CONSTANTS::PROBLEM\_-OPTIMISATION\_CLASS.

Here is the call graph for this function:

```
6.38.2.30 subroutine PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_NUMBER
 (INTEGER(INTG),intent(in) USER_NUMBER, INTEGER(INTG),intent(in)
 PROBLEM_CLASS, INTEGER(INTG),intent(in) PROBLEM_EQUATION_TYPE,
 INTEGER(INTG),intent(in) PROBLEM_SUBTYPE, INTEGER(INTG),intent(out)
 ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Sets/changes the problem specification i.e., problem class, type and subtype for a problem identified by a user number.

**Parameters:**

**USER\_NUMBER** The user number of the problem to set the specification for.  
**PROBLEM\_CLASS** The problem class to set.  
**PROBLEM\_EQUATION\_TYPE** The problem equation to set.  
**PROBLEM\_SUBTYPE** The problem subtype.  
**ERR** The error code  
**ERROR** The error string

Definition at line 1430 of file problem\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `PROBLEM_USER_NUMBER_FIND()`.

Here is the call graph for this function:

**6.38.2.31 subroutine PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_SET\_PTR**  
`(TYPE(PROBLEM_TYPE),pointer PROBLEM, INTEGER(INTG),intent(in)`  
`PROBLEM_CLASS, INTEGER(INTG),intent(in) PROBLEM_EQUATION_TYPE,`  
`INTEGER(INTG),intent(in) PROBLEM_SUBTYPE, INTEGER(INTG),intent(out)`  
`ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Sets/changes the problem specification i.e., problem class, type and subtype for a problem identified by a pointer.

**Parameters:**

**PROBLEM** A pointer to the problem to set the specification for.  
**PROBLEM\_CLASS** The problem class to set.  
**PROBLEM\_EQUATION\_TYPE** The problem equation type to set.  
**PROBLEM\_SUBTYPE** The problem subtype to set.  
**ERR** The error code  
**ERROR** The error string

Definition at line 1459 of file problem\_routines.f90.

References    `CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_SET()`,    `ELASTICITY_ROUTINES::ELASTICITY_PROBLEM_CLASS_TYPE_SET()`,    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `PROBLEM_CONSTANTS::PROBLEM_CLASSICAL_FIELD_CLASS`,    `PROBLEM_CONSTANTS::PROBLEM_ELASTICITY_CLASS`,    `PROBLEM_CONSTANTS::PROBLEM_ELECTROMAGNETICS_CLASS`,    `PROBLEM_CONSTANTS::PROBLEM_FITTING_CLASS`,    `PROBLEM_CONSTANTS::PROBLEM_FLUID_MECHANICS_CLASS`,    `PROBLEM_CONSTANTS::PROBLEM_MODAL_CLASS`,    and `PROBLEM_CONSTANTS::PROBLEM_OPTIMISATION_CLASS`.

Here is the call graph for this function:

**6.38.2.32 subroutine PROBLEM\_ROUTINES::PROBLEM\_USER\_NUMBER\_FIND**  
`(INTEGER(INTG),intent(in) USER_NUMBER, TYPE(PROBLEM_TYPE),pointer PROBLEM, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Finds and returns in PROBLEM a pointer to the problem identified by USER\_NUMBER. If no problem with that USER\_NUMBER exists PROBLEM is left nullified.

**Parameters:**

**USER\_NUMBER** The user number to find.

**PROBLEM** On return a pointer to the problem with the given user number. If no problem with that user number exists then the pointer is returned as NULL. Must not be associated on entry.

**ERR** The error code

**ERROR** The error string

Definition at line 1513 of file problem\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), PROBLEMS, and KINDS::PTR.

Referenced by PROBLEM\_CREATE\_START(), PROBLEM\_SPECIFICATION\_GET\_NUMBER(), and PROBLEM\_SPECIFICATION\_SET\_NUMBER().

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.38.2.33 subroutine PROBLEM\_ROUTINES::PROBLEMS\_FINALISE (INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

Finalises all problems and deallocates all memory.

**Parameters:**

**ERR** The error code

**ERROR** The error string

Definition at line 1550 of file problem\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), PROBLEMS, and KINDS::PTR.

Referenced by CMISS::CMISS\_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.38.2.34 subroutine PROBLEM\_ROUTINES::PROBLEMS\_INITIALISE (INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

Initialises all problems.

**Parameters:**

**ERR** The error code

**ERROR** The error string

Definition at line 1577 of file problem\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), and PROBLEMS.

Referenced by CMISS::CMISS\_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

### 6.38.3 Variable Documentation

#### 6.38.3.1 TYPE(PROBLEMS\_TYPE),target PROBLEM\_ROUTINES::PROBLEMS

Definition at line 94 of file problem\_routines.f90.

Referenced by PROBLEM\_CREATE\_FINISH(), PROBLEM\_CREATE\_START(), PROBLEM\_DESTROY\_NUMBER(), PROBLEM\_DESTROY\_PTR(), PROBLEM\_INITIALISE(), PROBLEM\_USER\_NUMBER\_FIND(), PROBLEMS\_FINALISE(), and PROBLEMS\_INITIALISE().

## 6.39 REGION\_ROUTINES Namespace Reference

This module contains all region routines.

### Classes

- interface [REGION\\_COORDINATE\\_SYSTEM\\_SET](#)
- interface [REGION\\_LABEL\\_SET](#)

### Functions

- `TYPE(COORDINATE_SYSTEM_TYPE) REGION_COORDINATE_SYSTEM_GET (REGION, ERR, ERROR)`
- subroutine `REGION_COORDINATE_SYSTEM_SET_NUMBER (USER_NUMBER, COORDINATE_SYSTEM, ERR, ERROR,*)`
- subroutine `REGION_COORDINATE_SYSTEM_SET_PTR (REGION, COORDINATE_SYSTEM, ERR, ERROR,*)`
- subroutine `REGION_CREATE_FINISH (REGION, ERR, ERROR,*)`
- subroutine `REGION_CREATE_START (USER_NUMBER, REGION, ERR, ERROR,*)`
- subroutine `REGION_DESTROY (USER_NUMBER, ERR, ERROR,*)`
- `TYPE(VARYING_STRING) REGION_LABEL_GET (REGION, ERR, ERROR)`
- subroutine `REGION_LABEL_SET_NUMBER (USER_NUMBER, LABEL, ERR, ERROR,*)`
- subroutine `REGION_LABEL_SET_PTR (REGION, LABEL, ERR, ERROR,*)`
- subroutine `REGION_SUB_REGION_CREATE_START (USER_NUMBER, PARENT_REGION, SUB_REGION, ERR, ERROR,*)`
- subroutine `REGION_SUB_REGION_CREATE_FINISH (REGION, ERR, ERROR,*)`
- subroutine `REGION_USER_NUMBER_FIND (USER_NUMBER, REGION, ERR, ERROR,*)`
- subroutine `REGION_USER_NUMBER_FIND_PTR (USER_NUMBER, REGION, START_REGION, ERR, ERROR,*)`
- subroutine `REGIONS_INITIALISE (ERR, ERROR,*)`
- subroutine `REGIONS_FINALISE (ERR, ERROR,*)`

### Variables

- `TYPE(REGION_TYPE), target GLOBAL_REGION`

#### 6.39.1 Detailed Description

This module contains all region routines.

#### 6.39.2 Function Documentation

##### 6.39.2.1 `TYPE(COORDINATE_SYSTEM_TYPE) REGION_ROUTINES::REGION_COORDINATE_SYSTEM_GET (TYPE(REGION_TYPE),pointer REGION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)`

Definition at line 92 of file region\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.39.2.2 subroutine REGION\_ROUTINES::REGION\_COORDINATE\_SYSTEM\_SET\_NUMBER**  
`(INTEGER(INTG),intent(in) USER_NUMBER,`  
`TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_SYSTEM,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`  
`[private]`

Definition at line 135 of file `region_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `REGION_COORDINATE_SYSTEM_SET_PTR()`, and `REGION_USER_NUMBER_FIND()`.

Here is the call graph for this function:

**6.39.2.3 subroutine REGION\_ROUTINES::REGION\_COORDINATE\_SYSTEM\_SET\_PTR**  
`(TYPE(REGION_TYPE),pointer REGION, TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_SYSTEM, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Definition at line 165 of file `region_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `REGION_COORDINATE_SYSTEM_SET_NUMBER()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.39.2.4 subroutine REGION\_ROUTINES::REGION\_CREATE\_FINISH**  
`(TYPE(REGION_TYPE),pointer REGION, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Definition at line 205 of file `region_routines.f90`.

References `BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE`, `BASE_ROUTINES::DIAGNOSTICS1`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `GLOBAL_REGION`, and `KINDS::PTR`.

Here is the call graph for this function:

**6.39.2.5 subroutine REGION\_ROUTINES::REGION\_CREATE\_START**  
`(INTEGER(INTG),intent(in) USER_NUMBER, TYPE(REGION_TYPE),pointer REGION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 249 of file `region_routines.f90`.

References `COORDINATE_ROUTINES::COORDINATE_SYSTEM_USER_NUMBER_FIND()`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

`FIELD_ROUTINES::FIELDS_INITIALISE()`, `GLOBAL_REGION`, `MESH_ROUTINES::MESHES_INITIALISE()`, `NODE_ROUTINES::NODES_INITIALISE()`, `KINDS::PTR`, and `REGION_USER_NUMBER_FIND()`.

Here is the call graph for this function:

**6.39.2.6 subroutine REGION\_ROUTINES::REGION\_DESTROY (INTEGER(INTG),intent(in) *USER\_NUMBER*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Definition at line 338 of file `region_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_ROUTINES::FIELDS_FINALISE()`, `MESH_ROUTINES::MESHES_FINALISE()`, `NODE_ROUTINES::NODES_FINALISE()`, `KINDS::PTR`, and `REGION_USER_NUMBER_FIND()`.

Referenced by `REGIONS_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.39.2.7 TYPE(VARYING\_STRING) REGION\_ROUTINES::REGION\_LABEL\_GET (TYPE(REGION\_TYPE),pointer *REGION*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*)**

Definition at line 418 of file `region_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.39.2.8 subroutine REGION\_ROUTINES::REGION\_LABEL\_SET\_NUMBER (INTEGER(INTG),intent(in) *USER\_NUMBER*, CHARACTER(LEN=\*),intent(in) *LABEL*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Definition at line 462 of file `region_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `REGION_LABEL_SET_PTR()`, and `REGION_USER_NUMBER_FIND()`.

Here is the call graph for this function:

**6.39.2.9 subroutine REGION\_ROUTINES::REGION\_LABEL\_SET\_PTR (TYPE(REGION\_TYPE),pointer *REGION*, CHARACTER(LEN=\*),intent(in) *LABEL*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Definition at line 492 of file `region_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `REGION_LABEL_SET_NUMBER()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.39.2.10 subroutine REGION\_ROUTINES::REGION\_SUB\_REGION\_CREATE\_FINISH  
 (TYPE(REGION\_TYPE),pointer *REGION*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Definition at line 622 of file region\_routines.f90.

References            BASE\_ROUTINES::DIAGNOSTIC\_OUTPUT\_TYPE,            BASE\_-ROUTINES::DIAGNOSTICS1,    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

**6.39.2.11 subroutine REGION\_ROUTINES::REGION\_SUB\_REGION\_CREATE\_START  
 (INTEGER(INTG),intent(in) *USER\_NUMBER*, TYPE(REGION\_TYPE),pointer  
*PARENT\_REGION*, TYPE(REGION\_TYPE),pointer *SUB\_REGION*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*)**

Definition at line 528 of file region\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(),    FIELD\_ROUTINES::FIELDS\_INITIALISE(),    MESH\_ROUTINES::MESHS\_INITIALISE(),    NODE\_ROUTINES::NODES\_INITIALISE(),    KINDS::PTR, and REGION\_USER\_-NUMBER\_FIND().

Here is the call graph for this function:

**6.39.2.12 subroutine REGION\_ROUTINES::REGION\_USER\_NUMBER\_FIND  
 (INTEGER(INTG),intent(in) *USER\_NUMBER*, TYPE(REGION\_TYPE),pointer  
*REGION*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out)  
*ERROR*, \*)**

Definition at line 667 of file region\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(),    GLOBAL\_REGION,    KINDS::PTR, and REGION\_USER\_NUMBER\_FIND\_-PTR().

Referenced by REGION\_COORDINATE\_SYSTEM\_SET\_NUMBER(), REGION\_CREATE\_START(), REGION\_DESTROY(), REGION\_LABEL\_SET\_NUMBER(), and REGION\_SUB\_REGION\_-CREATE\_START().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.39.2.13 subroutine REGION\_ROUTINES::REGION\_USER\_NUMBER\_FIND\_PTR**  
`(INTEGER(INTG),intent(in) USER_NUMBER, TYPE(REGION_TYPE),pointer  
 REGION, TYPE(REGION_TYPE),pointer START_REGION,  
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,  
 *) [private]`

Definition at line 706 of file region\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), and KINDS::PTR.

Referenced by REGION\_USER\_NUMBER\_FIND().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.39.2.14 subroutine REGION\_ROUTINES::REGIONS\_FINALISE**  
`(INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,  
 *)`

Definition at line 790 of file region\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), GLOBAL\_REGION, KINDS::PTR, and REGION\_DESTROY().

Referenced by CMISS::CMISS\_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.39.2.15 subroutine REGION\_ROUTINES::REGIONS\_INITIALISE**  
`(INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,  
 *)`

Definition at line 750 of file region\_routines.f90.

References    COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_USER\_NUMBER\_FIND(),  
 BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_ROUTINES::EXITS(), and  
 GLOBAL\_REGION.

Referenced by CMISS::CMISS\_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

### 6.39.3 Variable Documentation

#### 6.39.3.1 TYPE(REGION\_TYPE),target REGION\_ROUTINES::GLOBAL\_REGION

Definition at line 67 of file region\_routines.f90.

Referenced by REGION\_CREATE\_FINISH(), REGION\_CREATE\_START(), REGION\_USER\_NUMBER\_FIND(), REGIONS\_FINALISE(), and REGIONS\_INITIALISE().

## 6.40 SOLVER\_MATRICES\_ROUTINES Namespace Reference

This module handles all solver matrix and rhs routines.

### Functions

- subroutine [SOLVER\\_MATRICES\\_CREATE\\_FINISH](#) (SOLVER\_MATRICES, ERR, ERROR,\*)
 

*Finishes the process of creating the solver matrices.*
- subroutine [SOLVER\\_MATRICES\\_CREATE\\_START](#) (SOLVER, SOLVER\_MATRICES, ERR, ERROR,\*)
 

*Starts the process of creating the solver matrices.*
- subroutine [SOLVER\\_MATRICES\\_DESTROY](#) (SOLVER\_MATRICES, ERR, ERROR,\*)
 

*Destroys the solver matrices.*
- subroutine [SOLVER\\_MATRICES\\_FINALISE](#) (SOLVER\_MATRICES, ERR, ERROR,\*)
 

*Finalises the solver matrices and deallocates all memory.*
- subroutine [SOLVER\\_MATRICES\\_INITIALISE](#) (SOLVER, ERR, ERROR,\*)
 

*Initialises the solver matrices for a solver.*
- INTEGER(INTG) [SOLVER\\_MATRICES\\_LIBRARY\\_TYPE\\_GET](#) (SOLVER\_MATRICES, ERR, ERROR)
 

*Gets the library type for the solver matrices (and vectors).*
- subroutine [SOLVER\\_MATRICES\\_LIBRARY\\_TYPE\\_SET](#) (SOLVER\_MATRICES, LIBRARY\_TYPE, ERR, ERROR,\*)
 

*Sets the library type for the solver matrices (and vectors).*
- subroutine [SOLVER\\_MATRICES\\_OUTPUT](#) (ID, SOLVER\_MATRICES, ERR, ERROR,\*)
 

*Outputs the solver matrices.*
- subroutine [SOLVER\\_MATRICES\\_STORAGE\\_TYPE\\_GET](#) (SOLVER\_MATRICES, STORAGE\_TYPE, ERR, ERROR,\*)
 

*Gets the storage type (sparsity) of the solver matrices.*
- subroutine [SOLVER\\_MATRICES\\_STORAGE\\_TYPE\\_SET](#) (SOLVER\_MATRICES, STORAGE\_TYPE, ERR, ERROR,\*)
 

*Sets the storage type (sparsity) of the solver matrices.*
- subroutine [SOLVER\\_MATRIX\\_STRUCTURE\\_CALCULATE](#) (SOLVER\_MATRIX, NUMBER\_OF\_NON\_ZEROS, ROW\_INDICES, COLUMN\_INDICES, ERR, ERROR,\*)
 

*Calculates the structure (sparsity) of the solver matrix from the solution mapping.*
- subroutine [SOLVER\\_MATRIX\\_FINALISE](#) (SOLVER\_MATRIX, ERR, ERROR,\*)
 

*Finalises a solver matrix and deallocates all memory.*
- subroutine [SOLVER\\_MATRIX\\_FORM](#) (SOLVER\_MATRIX, ERR, ERROR,\*)
 

*Forms a solver matrix by initialising the structure of the matrix to zero.*

- subroutine **SOLVER\_MATRIX\_INITIALISE** (SOLVER\_MATRICES, MATRIX\_NUMBER, ERR, ERROR,\*)

*Initialises a solver matrix.*

### 6.40.1 Detailed Description

This module handles all solver matrix and rhs routines.

### 6.40.2 Function Documentation

#### 6.40.2.1 subroutine SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRICES\_CREATE\_FINISH (TYPE(SOLVER\_MATRICES\_TYPE),pointer SOLVER\_MATRICES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)

Finishes the process of creating the solver matrices.

**Parameters:**

**SOLVER\_MATRICES** A pointer to the solver matrices

**ERR** The error code

**ERROR** The error string

Definition at line 77 of file solver\_matrices\_routines.f90.

References **BASE\_ROUTINES::ENTERS()**, **BASE\_ROUTINES::ERRORS()**, **BASE\_ROUTINES::EXITS()**, **MATRIX\_VECTOR::MATRIX\_BLOCK\_STORAGE\_TYPE**, **MATRIX\_VECTOR::MATRIX\_VECTOR\_DP\_TYPE**, **PROBLEM\_CONSTANTS::PROBLEM SOLUTION\_NONLINEAR**, **KINDS::PTR**, **SOLVER\_MATRICES\_FINALISE()**, and **SOLVER\_MATRIX\_STRUCTURE\_CALCULATE()**.

Referenced by **SOLVER\_ROUTINES::SOLVER\_LINEAR\_DIRECT\_CREATE\_FINISH()**, **SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_CREATE\_FINISH()**, **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_CREATE\_FINISH()**, and **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_TRUSTREGION\_CREATE\_FINISH()**.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.40.2.2 subroutine SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRICES\_CREATE\_START (TYPE(SOLVER\_TYPE),pointer SOLVER, TYPE(SOLVER\_MATRICES\_TYPE),pointer SOLVER\_MATRICES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]

Starts the process of creating the solver matrices.

**Parameters:**

**SOLVER** A pointer to the solver to create the solver matrices for

**SOLVER\_MATRICES** A pointer to the solver matrices

**ERR** The error code

**ERROR** The error string

Definition at line 195 of file solver\_matrices\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `SOLVER_MATRICES_FINALISE()`, and `SOLVER_MATRICES_INITIALISE()`.

Referenced    by    `SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_CREATE_FINISH()`,  
`SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()`,    `SOLVER_ROUTINES::SOLVER_NONLINEAR_LINESEARCH_CREATE_FINISH()`,    and    `SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.40.2.3 subroutine `SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_DESTROY` `(TYPE(SOLVER_MATRICES_TYPE),pointer SOLVER_MATRICES,` `INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Destroy the solver matrices.

##### Parameters:

**SOLVER\_MATRICES** A pointer the solver matrices to destroy

**ERR** The error code

**ERROR** The error string

Definition at line 238 of file solver\_matrices\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `SOLVER_MATRICES_FINALISE()`.

Here is the call graph for this function:

#### 6.40.2.4 subroutine `SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_FINALISE` `(TYPE(SOLVER_MATRICES_TYPE),pointer SOLVER_MATRICES,` `INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)` `[private]`

Finalises the solver matrices and deallocates all memory.

##### Parameters:

**SOLVER\_MATRICES** A pointer to the solver matrices to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 267 of file solver\_matrices\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `KINDS::PTR`, and `SOLVER_MATRIX_FINALISE()`.

Referenced by `SOLVER_MATRICES_CREATE_FINISH()`, `SOLVER_MATRICES_CREATE_START()`, `SOLVER_MATRICES_DESTROY()`, and `SOLVER_MATRICES_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.40.2.5 subroutine SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRICES\_INITIALISE  
(TYPE(SOLVER\_TYPE),pointer *SOLVER*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Initialises the solver matrices for a solver.

**Parameters:**

***SOLVER*** A pointer to the problem solver to initialise the solver matrices for  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 303 of file solver\_matrices\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    **BASE\_ROUTINES::EXITS()**,    **PROBLEM\_CONSTANTS::PROBLEM SOLUTION\_NONLINEAR**,    **KINDS::PTR**, **SOLVER\_MATRICES\_FINALISE()**, and **SOLVER\_MATRIX\_INITIALISE()**.

Referenced by **SOLVER\_MATRICES\_CREATE\_START()**.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.40.2.6 INTEGER(INTG) SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRICES\_-  
LIBRARY\_TYPE\_GET (TYPE(SOLVER\_MATRICES\_TYPE),pointer  
*SOLVER\_MATRICES*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]**

Gets the library type for the solver matrices (and vectors).

**Parameters:**

***SOLVER\_MATRICES*** A pointer to the solver matrices.  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 391 of file solver\_matrices\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    and    **BASE\_ROUTINES::EXITS()**.

Here is the call graph for this function:

**6.40.2.7 subroutine SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRICES\_LIBRARY\_-  
TYPE\_SET (TYPE(SOLVER\_MATRICES\_TYPE),pointer *SOLVER\_MATRICES*,  
INTEGER(INTG),intent(in) *LIBRARY\_TYPE*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Sets the library type for the solver matrices (and vectors).

**Parameters:**

**SOLVER\_MATRICES** A pointer to the solver matrices.

**LIBRARY\_TYPE** The library type to set

See also:

DISTRIBUTED\_MATRIX\_VECTOR\_LibraryTypes

**ERR** The error code

**ERROR** The error string

Definition at line 422 of file solver\_matrices\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by SOLVER\_ROUTINES::SOLVER\_LINEAR\_DIRECT\_CREATE\_FINISH(), SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_CREATE\_FINISH(), SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_CREATE\_FINISH(), and SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_TRUSTREGION\_CREATE\_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.40.2.8 subroutine SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRICES\_OUTPUT (INTEGER(INTG),intent(in) ID, TYPE(SOLVER\_MATRICES\_TYPE),pointer SOLVER\_MATRICES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Outputs the solver matrices.

**Parameters:**

**ID** The ID of the ouput stream

**SOLVER\_MATRICES** A pointer to the solver matrices

**ERR** The error code

**ERROR** The error string

Definition at line 464 of file solver\_matrices\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and KINDS::PTR.

Referenced by SOLVER\_ROUTINES::SOLVER\_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.40.2.9 subroutine SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRICES\_STORAGE\_TYPE\_GET (TYPE(SOLVER\_MATRICES\_TYPE),pointer SOLVER\_MATRICES, INTEGER(INTG),dimension(:),pointer STORAGE\_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \* ) [private]**

Gets the storage type (sparsity) of the solver matrices.

**Parameters:**

**SOLVER\_MATRICES** A pointer to the solver matrices

**STORAGE\_TYPE** STORAGE\_TYPE(matrix\_idx). The storage type for the matrix\_idx'th solver matrix

**ERR** The error code

**ERROR** The error string

Definition at line 519 of file solver\_matrices\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

```
6.40.2.10 subroutine SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_-
 STORAGE_TYPE_SET (TYPE(SOLVER_MATRICES_TYPE),pointer
 SOLVER_MATRICES, INTEGER(INTG),dimension(:),intent(in) STORAGE_TYPE,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *)
```

Sets the storage type (sparsity) of the solver matrices.

**Parameters:**

**SOLVER\_MATRICES** A pointer to the solver matrices

**STORAGE\_TYPE** STORAGE\_TYPE(matrix\_idx). The storage type for the matrix\_idx'th solver matrix

**ERR** The error code

**ERROR** The error string

Definition at line 563 of file solver\_matrices\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), MATRIX\_VECTOR::MATRIX\_BLOCK\_STORAGE\_TYPE, MATRIX\_VECTOR::MATRIX\_COLUMN\_MAJOR\_STORAGE\_TYPE, MATRIX\_VECTOR::MATRIX\_COMPRESSED\_COLUMN\_STORAGE\_TYPE, MATRIX\_VECTOR::MATRIX\_COMPRESSED\_ROW\_STORAGE\_TYPE, MATRIX\_VECTOR::MATRIX\_DIAGONAL\_STORAGE\_TYPE, MATRIX\_VECTOR::MATRIX\_ROW\_COLUMN\_STORAGE\_TYPE, MATRIX\_VECTOR::MATRIX\_ROW\_MAJOR\_STORAGE\_TYPE, and KINDS::PTR.

Referenced by SOLVER\_ROUTINES::SOLVER\_LINEAR\_DIRECT\_CREATE\_FINISH(), SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_CREATE\_FINISH(), and SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_CREATE\_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

```
6.40.2.11 subroutine SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_FINALISE
 (TYPE(SOLVER_MATRIX_TYPE),pointer SOLVER_MATRIX,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *) [private]
```

Finalises a solver matrix and deallocates all memory.

**Parameters:**

**SOLVER\_MATRIX** A pointer to the solver matrix to finalise  
**ERR** The error code  
**ERROR** The error string

Definition at line 1013 of file solver\_matrices\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_MATRICES_FINALISE()`, and `SOLVER_MATRIX_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.40.2.12 subroutine SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRIX\_FORM**  
`(TYPE(SOLVER_MATRIX_TYPE),pointer SOLVER_MATRIX,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`  
`*) [private]`

Forms a solver matrix by initialising the strucuture of the matrix to zero.

**Parameters:**

**SOLVER\_MATRIX** A pointer to the solver matrix to finalise  
**ERR** The error code  
**ERROR** The error string

Definition at line 1042 of file solver\_matrices\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.40.2.13 subroutine SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRIX\_INITIALISE**  
`(TYPE(SOLVER_MATRICES_TYPE),pointer SOLVER_MATRICES,`  
`INTEGER(INTG),intent(in) MATRIX_NUMBER, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Initialises a solver matrix.

**Parameters:**

**SOLVER\_MATRICES** A pointer to the solver matrices to initialise  
**MATRIX\_NUMBER** The matrix number in the solver matrices to initialise  
**ERR** The error code  
**ERROR** The error string

Definition at line 1071 of file solver\_matrices\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_VECTOR::MATRIX_BLOCK_STORAGE_TYPE`, `KINDS::PTR`, and `SOLVER_MATRIX_FINALISE()`.

Referenced by SOLVER\_MATRICES\_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

```
6.40.2.14 subroutine SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_-
STRUCTURE_CALCULATE (TYPE(SOLVER_MATRIX_TYPE),pointer
SOLVER_MATRIX, INTEGER(INTG),intent(out) NUMBER_OF_-
NON_ZEROS, INTEGER(INTG),dimension(:,),pointer ROW_INDICES,
INTEGER(INTG),dimension(:,),pointer COLUMN_INDICES,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 $\ast)$ [private]
```

Calculates the structure (sparsity) of the solver matrix from the soluton mapping.

#### Parameters:

**SOLVER\_MATRIX** A pointer to the solver matrix to calculate the structure for  
**NUMBER\_OF\_NON\_ZEROS** On return the number of non-zeros in the solver matrix  
**ROW\_INDICES** On return a pointer to row location indices in compressed row format. The pointers must be NULL on entry and the calling routine is responsible for deallocation.  
**COLUMN\_INDICES** On return a pointer to the column location indices in compressed row format. The pointers must be NULL on entry and the calling routine is responsible for deallocation.  
**ERR** The error code  
**ERROR** The error string

Definition at line 632 of file solver\_matrices\_routines.f90.

References            `BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE, BASE_-ROUTINES::DIAGNOSTICS1, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), LISTS::LIST_CREATE_FINISH(), LISTS::LIST_CREATE_START(), LISTS::LIST_DATA_TYPE_SET(), LISTS::LIST_DESTROY(), LISTS::LIST_INITIAL_SIZE_SET(), LISTS::LIST_INTG_TYPE, LISTS::LIST_NUMBER_OF_ITEMS_GET(), LISTS::LIST_REMOVE_DUPLICATES(), and KINDS::PTR.`

Referenced by SOLVER\_MATRICES\_CREATE\_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

## 6.41 SOLVER\_ROUTINES Namespace Reference

This module handles all solver routines.

### Functions

- subroutine [SOLVER\\_CREATE\\_FINISH](#) (SOLVER, ERR, ERROR,\*)
 

*Finishes the process of creating a solver for a problem solution.*
- subroutine [SOLVER\\_CREATE\\_START](#) (SOLUTION, SOLVE\_TYPE, SOLVER, ERR, ERROR,\*)
 

*Starts the process of creating a solver for a problem solution.*
- subroutine [SOLVER\\_DESTROY](#) (SOLVER, ERR, ERROR,\*)
 

*Destroy a problem solver.*
- subroutine [SOLVER\\_EIGENPROBLEM\\_CREATE\\_FINISH](#) (EIGENPROBLEM\_SOLVER, ERR, ERROR,\*)
 

*Finishes the process of creating a eigenproblem solver.*
- subroutine [SOLVER\\_EIGENPROBLEM\\_FINALISE](#) (EIGENPROBLEM\_SOLVER, ERR, ERROR,\*)
 

*Finalise a eigenproblem solver for a problem solver.*
- subroutine [SOLVER\\_EIGENPROBLEM\\_INITIALISE](#) (SOLVER, ERR, ERROR,\*)
 

*Initialise a eigenproblem solver for a problem solver.*
- subroutine [SOLVER\\_EIGENPROBLEM\\_SOLVE](#) (EIGENPROBLEM\_SOLVER, ERR, ERROR,\*)
 

*Solve a eigenproblem solver.*
- subroutine [SOLVER\\_FINALISE](#) (SOLVER, ERR, ERROR,\*)
 

*Finalises a problem solver and deallocates all memory.*
- subroutine [SOLVER\\_INITIALISE](#) (SOLUTION, SOLVE\_TYPE, ERR, ERROR,\*)
 

*Initialise a solver for a problem solution.*
- subroutine [SOLVER\\_LIBRARY\\_SET](#) (SOLVER, SOLVER\_LIBRARY, ERR, ERROR,\*)
 

*Sets/changes the type of library to use for the solver.*
- subroutine [SOLVER\\_LINEAR\\_CREATE\\_FINISH](#) (LINEAR\_SOLVER, ERR, ERROR,\*)
 

*Finishes the process of creating a linear solver.*
- subroutine [SOLVER\\_LINEAR\\_DIRECT\\_CREATE\\_FINISH](#) (LINEAR\_DIRECT\_SOLVER, ERR, ERROR,\*)
 

*Finishes the process of creating a linear direct solver.*
- subroutine [SOLVER\\_LINEAR\\_DIRECT\\_FINALISE](#) (LINEAR\_DIRECT\_SOLVER, ERR, ERROR,\*)
 

*Finalise a direct linear solver for a linear solver and deallocate all memory.*

- subroutine **SOLVER\_LINEAR\_DIRECT\_INITIALISE** (LINEAR\_SOLVER, ERR, ERROR,\*)
 

*Initialise a direct linear solver for a lienar solver.*
- subroutine **SOLVER\_LINEAR\_DIRECT\_SOLVE** (LINEAR\_DIRECT\_SOLVER, ERR, ERROR,\*)
 

*Solve a linear direct solver.*
- subroutine **SOLVER\_LINEAR\_DIRECT\_TYPE\_SET** (SOLVER, DIRECT\_SOLVER\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the type of direct linear solver.*
- subroutine **SOLVER\_LINEAR\_FINALISE** (LINEAR\_SOLVER, ERR, ERROR,\*)
 

*Finalise a linear solver for a problem solver.*
- subroutine **SOLVER\_LINEAR\_INITIALISE** (SOLVER, ERR, ERROR,\*)
 

*Initialise a linear solver for a problem solver.*
- subroutine **SOLVER\_LINEAR\_ITERATIVE\_ABSOLUTE\_TOLERANCE\_SET** (SOLVER, ABSOLUTE\_TOLERANCE, ERR, ERROR,\*)
 

*Sets/changes the maximum absolute tolerance for an iterative linear solver.*
- subroutine **SOLVER\_LINEAR\_ITERATIVE\_CREATE\_FINISH** (LINEAR\_ITERATIVE\_SOLVER, ERR, ERROR,\*)
 

*Finishes the process of creating a linear iterative solver.*
- subroutine **SOLVER\_LINEAR\_ITERATIVE\_DIVERGENCE\_TOLERANCE\_SET** (SOLVER, DIVERGENCE\_TOLERANCE, ERR, ERROR,\*)
 

*Sets/changes the maximum divergence tolerance for an iterative linear solver.*
- subroutine **SOLVER\_LINEAR\_ITERATIVE\_FINALISE** (LINEAR\_ITERATIVE\_SOLVER, ERR, ERROR,\*)
 

*Finalise an iterative linear solver for a linear solver and deallocate all memory.*
- subroutine **SOLVER\_LINEAR\_ITERATIVE\_INITIALISE** (LINEAR\_SOLVER, ERR, ERROR,\*)
 

*Initialise an iterative linear solver for a linear solver.*
- subroutine **SOLVER\_LINEAR\_ITERATIVE\_MAXIMUM\_ITERATIONS\_SET** (SOLVER, MAXIMUM\_ITERATIONS, ERR, ERROR,\*)
 

*Sets/changes the maximum number of iterations for an iterative linear solver.*
- subroutine **SOLVER\_LINEAR\_ITERATIVE\_PRECONDITIONER\_TYPE\_SET** (SOLVER, ITERATIVE\_PRECONDITIONER\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the type of preconditioner for an iterative linear solver.*
- subroutine **SOLVER\_LINEAR\_ITERATIVE\_RELATIVE\_TOLERANCE\_SET** (SOLVER, RELATIVE\_TOLERANCE, ERR, ERROR,\*)
 

*Sets/changes the relative tolerance for an iterative linear solver.*

- subroutine **SOLVER\_LINEAR\_ITERATIVE\_SOLVE** (LINEAR\_ITERATIVE\_SOLVER, ERR, ERROR,\*)
 

*Solves a linear iterative linear solver.*
- subroutine **SOLVER\_LINEAR\_ITERATIVE\_TYPE\_SET** (SOLVER, ITERATIVE\_SOLVER\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the type of iterative linear solver.*
- subroutine **SOLVER\_LINEAR\_SOLVE** (LINEAR\_SOLVER, ERR, ERROR,\*)
 

*Solve a linear solver.*
- subroutine **SOLVER\_LINEAR\_TYPE\_SET** (SOLVER, LINEAR\_SOLVE\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the type of linear solver.*
- subroutine **SOLVER\_MATRICES\_ASSEMBLE** (SOLVER, ERR, ERROR,\*)
 

*Assembles the solver matrices and rhs from the equations.*
- subroutine **SOLVER\_NONLINEAR\_ABSOLUTE\_TOLERANCE\_SET** (SOLVER, ABSOLUTE\_TOLERANCE, ERR, ERROR,\*)
 

*Sets/changes the maximum absolute tolerance for a nonlinear solver.*
- subroutine **SOLVER\_NONLINEAR\_CREATE\_FINISH** (NONLINEAR\_SOLVER, ERR, ERROR,\*)
 

*Finishes the process of creating a nonlinear solver.*
- subroutine **SOLVER\_NONLINEAR\_FINALISE** (NONLINEAR\_SOLVER, ERR, ERROR,\*)
 

*Finalise a nonlinear solver for a problem solver.*
- subroutine **SOLVER\_NONLINEAR\_INITIALISE** (SOLVER, ERR, ERROR,\*)
 

*Initialise a nonlinear solver for a problem solver.*
- subroutine **SOLVER\_NONLINEAR\_JACOBIAN\_EVALUATE** (SOLVER, ERR, ERROR,\*)
 

*Evaluates the Jacobian for a Newton like nonlinear solver.*
- subroutine **SOLVER\_NONLINEAR\_JACOBIAN\_EVALUATE\_PETSC** (SNES, X, A, B, FLAG, CTX)
 

*Called from the PETSc SNES solvers to evaluate the Jacobian for a Newton like nonlinear solver.*
- subroutine **SOLVER\_NONLINEAR\_LINESEARCH\_ALPHA\_SET** (SOLVER, LINESEARCH\_ALPHA, ERR, ERROR,\*)
 

*Sets/changes the line search alpha for a nonlinear solver.*
- subroutine **SOLVER\_NONLINEAR\_LINESEARCH\_CREATE\_FINISH** (NONLINEAR\_LINESEARCH\_SOLVER, ERR, ERROR,\*)
 

*Finishes the process of creating nonlinear Newton line search solver.*
- subroutine **SOLVER\_NONLINEAR\_LINESEARCH\_FINALISE** (NONLINEAR\_LINESEARCH\_SOLVER, ERR, ERROR,\*)
 

*Finalise a nonlinear Newton line search solver and deallocate all memory.*

- subroutine **SOLVER\_NONLINEAR\_LINESearch\_INITIALISE** (NONLINEAR\_SOLVER, ERR, ERROR,\*)
 

*Initialise a nonlinear Newton line search solver for a problem solver.*
- subroutine **SOLVER\_NONLINEAR\_LINESearch\_MAXSTEP\_SET** (SOLVER, LINESEARCH\_MAXSTEP, ERR, ERROR,\*)
 

*Sets/changes the line search maximum step for a nonlinear solver.*
- subroutine **SOLVER\_NONLINEAR\_LINESearch\_solve** (NONLINEAR\_LINESearch\_- SOLVER, ERR, ERROR,\*)
 

*Sets/changes the line search step tolerance for a nonlinear solver.*
- subroutine **SOLVER\_NONLINEAR\_LINESearch\_STEPTOL\_SET** (SOLVER, LINESEARCH\_STEPTOL, ERR, ERROR,\*)
 

*Sets/changes the line search type for a nonlinear solver.*
- subroutine **SOLVER\_NONLINEAR\_MAXIMUM\_FUNCTION\_EVALUATIONS\_SET** (SOLVER, MAXIMUM\_FUNCTION\_EVALUATIONS, ERR, ERROR,\*)
 

*Sets/changes the maximum number of function evaluations for a nonlinear solver.*
- subroutine **SOLVER\_NONLINEAR\_MAXIMUM\_ITERATIONS\_SET** (SOLVER, MAXIMUM\_- ITERATIONS, ERR, ERROR,\*)
 

*Sets/changes the maximum number of iterations for a nonlinear solver.*
- subroutine **SOLVER\_NONLINEAR\_RELATIVE\_TOLERANCE\_SET** (SOLVER, RELATIVE\_- TOLERANCE, ERR, ERROR,\*)
 

*Sets/changes the relative tolerance for a nonlinear solver.*
- subroutine **SOLVER\_NONLINEAR\_RESIDUAL\_EVALUATE** (SOLVER, ERR, ERROR,\*)
 

*Evaluates the residual for a Newton like nonlinear solver.*
- subroutine **SOLVER\_NONLINEAR\_RESIDUAL\_EVALUATE\_PETSC** (SNES, X, F, CTX)
 

*Called from the PETSc SNES solvers to evaluate the residual for a Newton like nonlinear solver.*
- subroutine **SOLVER\_NONLINEAR\_SOLUTION\_TOLERANCE\_SET** (SOLVER, SOLUTION\_- TOLERANCE, ERR, ERROR,\*)
 

*Sets/changes the solution tolerance for a nonlinear solver.*
- subroutine **SOLVER\_NONLINEAR\_solve** (NONLINEAR\_SOLVER, ERR, ERROR,\*)
 

*Sets/changes the trust region delta0 for a nonlinear solver.*
- subroutine **SOLVER\_NONLINEAR\_TRUSTREGION\_CREATE\_FINISH** (NONLINEAR\_- TRUSTREGION\_SOLVER, ERR, ERROR,\*)
 

*Finishes the process of creating nonlinear trust region solver.*
- subroutine **SOLVER\_NONLINEAR\_TRUSTREGION\_DELTA0\_SET** (SOLVER, TRUSTREGION\_DELTA0, ERR, ERROR,\*)
 

*Sets/changes the trust region delta0 for a nonlinear solver.*

- subroutine **SOLVER\_NONLINEAR\_TRUSTREGION\_FINALISE** (NONLINEAR\_-  
TRUSTREGION\_SOLVER, ERR, ERROR,\*)  
*Finalise a nonlinear trust region solver and deallocate all memory.*
- subroutine **SOLVER\_NONLINEAR\_TRUSTREGION\_INITIALISE** (NONLINEAR\_SOLVER,  
ERR, ERROR,\*)  
*Initialise a nonlinear trust region solver for a problem solver.*
- subroutine **SOLVER\_NONLINEAR\_TRUSTREGION\_SOLVE** (NONLINEAR\_-  
TRUSTREGION\_SOLVER, ERR, ERROR,\*)  
• subroutine **SOLVER\_NONLINEAR\_TRUSTREGION\_TOLERANCE\_SET** (SOLVER,  
TRUSTREGION\_TOLERANCE, ERR, ERROR,\*)  
*Sets/changes the trust region tolerance for a nonlinear solver.*
- subroutine **SOLVER\_NONLINEAR\_TYPE\_SET** (SOLVER, NONLINEAR\_SOLVE\_TYPE, ERR,  
ERROR,\*)  
*Sets/changes the type of nonlinear solver.*
- subroutine **SOLVER\_OUTPUT\_TYPE\_SET** (SOLVER, OUTPUT\_TYPE, ERR, ERROR,\*)  
*Sets/changes the output type for a solver.*
- subroutine **SOLVER\_SPARSITY\_TYPE\_SET** (SOLVER, SPARSITY\_TYPE, ERR, ERROR,\*)  
*Sets/changes the sparsity type for a solver.*
- subroutine **SOLVER\_TIME\_INTEGRATION\_CREATE\_FINISH** (TIME\_INTEGRATION\_-  
SOLVER, ERR, ERROR,\*)  
*Finishes the process of creating a time integration solver.*
- subroutine **SOLVER\_TIME\_INTEGRATION\_FINALISE** (TIME\_INTEGRATION\_SOLVER,  
ERR, ERROR,\*)  
*Finalise a time integration solver for a problem solver.*
- subroutine **SOLVER\_TIME\_INTEGRATION\_INITIALISE** (SOLVER, ERR, ERROR,\*)  
*Initialise a time integration solver for a problem solver.*
- subroutine **SOLVER\_TIME\_INTEGRATION\_SOLVE** (TIME\_INTEGRATION\_SOLVER, ERR,  
ERROR,\*)  
*Solve a time integration solver.*
- subroutine **SOLVER\_SOLVE** (SOLVER, ERR, ERROR,\*)  
*Solve the problem.*
- subroutine **SOLVER\_VARIABLES\_UPDATE** (SOLVER, ERR, ERROR,\*)  
*Updates the dependent variables from the solver solution.*

## Variables

- INTEGER(INTG), parameter **SOLVER\_LINEAR\_TYPE** = 1  
*Linear solution solver.*

- INTEGER(INTG), parameter **SOLVER\_NONLINEAR\_TYPE** = 2  
*A nonlinear solution solver.*
- INTEGER(INTG), parameter **SOLVER\_TIME\_INTEGRATION\_TYPE** = 3  
*A time integration solver.*
- INTEGER(INTG), parameter **SOLVER\_EIGENPROBLEM\_TYPE** = 4  
*A eigenproblem type.*
- INTEGER(INTG), parameter **SOLVER\_CMISS\_LIBRARY** = LIBRARY\_CMISS\_TYPE  
*CMISS (internal) solver library.*
- INTEGER(INTG), parameter **SOLVER\_PETSC\_LIBRARY** = LIBRARY\_PETSC\_TYPE  
*PETSc solver library.*
- INTEGER(INTG), parameter **SOLVER\_LINEAR\_DIRECT\_SOLVE\_TYPE** = 1  
*Direct linear solver type.*
- INTEGER(INTG), parameter **SOLVER\_LINEAR\_ITERATIVE\_SOLVE\_TYPE** = 2  
*Iterative linear solver type.*
- INTEGER(INTG), parameter **SOLVER\_DIRECT\_LU** = 1  
*LU direct linear solver.*
- INTEGER(INTG), parameter **SOLVER\_DIRECT\_CHOLESKY** = 2  
*Cholesky direct linear solver.*
- INTEGER(INTG), parameter **SOLVER\_DIRECT\_SVD** = 3  
*SVD direct linear solver.*
- INTEGER(INTG), parameter **SOLVER\_ITERATIVE\_RICHARDSON** = 1  
*Richardson iterative solver type.*
- INTEGER(INTG), parameter **SOLVER\_ITERATIVE\_CHEBYCHEV** = 2  
*Chebychev iterative solver type.*
- INTEGER(INTG), parameter **SOLVER\_ITERATIVE\_CONJUGATE\_GRADIENT** = 3  
*Conjugate gradient iterative solver type.*
- INTEGER(INTG), parameter **SOLVER\_ITERATIVE\_BICONJUGATE\_GRADIENT** = 4  
*Bi-conjugate gradient iterative solver type.*
- INTEGER(INTG), parameter **SOLVER\_ITERATIVE\_GMRES** = 5  
*Generalised minimum residual iterative solver type.*
- INTEGER(INTG), parameter **SOLVER\_ITERATIVE\_BICGSTAB** = 6  
*Stabalised bi-conjugate gradient iterative solver type.*
- INTEGER(INTG), parameter **SOLVER\_ITERATIVE\_CONJGRAD\_SQUARED** = 7

*Conjugate gradient squared iterative solver type.*

- INTEGER(INTG), parameter **SOLVER\_ITERATIVE\_NO\_PRECONDITIONER** = 0  
*No preconditioner type.*
- INTEGER(INTG), parameter **SOLVER\_ITERATIVE\_JACOBI\_PRECONDITIONER** = 1  
*Jacobi preconditioner type.*
- INTEGER(INTG), parameter **SOLVER\_ITERATIVE\_BLOCK\_JACOBI\_PRECONDITIONER** = 2  
*Iterative block Jacobi preconditioner type.*
- INTEGER(INTG), parameter **SOLVER\_ITERATIVE\_SOR\_PRECONDITIONER** = 3  
*Successive over relaxation preconditioner type.*
- INTEGER(INTG), parameter **SOLVER\_ITERATIVE\_INCOMPLETE\_CHOLESKY\_PRECONDITIONER** = 4  
*Incomplete Cholesky preconditioner type.*
- INTEGER(INTG), parameter **SOLVER\_ITERATIVE\_INCOMPLETE\_LU\_PRECONDITIONER** = 5  
*Incomplete LU preconditioner type.*
- INTEGER(INTG), parameter **SOLVER\_ITERATIVE\_ADDITIVE\_SCHWARZ\_PRECONDITIONER** = 6  
*Additive Schwarz preconditioner type.*
- INTEGER(INTG), parameter **SOLVER\_NONLINEAR\_LINESEARCH** = 1  
*Newton line search nonlinear solver type.*
- INTEGER(INTG), parameter **SOLVER\_NONLINEAR\_TRUSTREGION** = 2  
*Newton trust region nonlinear solver type.*
- INTEGER(INTG), parameter **SOLVER\_NONLINEAR\_NONORMS\_LINESEARCH** = 1  
*No norms line search for Newton line search nonlinear solves.*
- INTEGER(INTG), parameter **SOLVER\_NONLINEAR\_NO\_LINESEARCH** = 2  
*No line search for Newton line search nonlinear solves.*
- INTEGER(INTG), parameter **SOLVER\_NONLINEAR\_QUADRATIC\_LINESEARCH** = 3  
*Quadratic search for Newton line search nonlinear solves.*
- INTEGER(INTG), parameter **SOLVER\_NONLINEAR\_CUBIC\_LINESEARCH** = 4  
*Cubic search for Newton line search nonlinear solves.*
- INTEGER(INTG), parameter **SOLVER\_NONLINEAR\_JACOBIAN\_NOT\_CALCULATED** = 1  
*The Jacobian values will not be calculated for the nonlinear equations set.*
- INTEGER(INTG), parameter **SOLVER\_NONLINEAR\_JACOBIAN\_ANALYTIC\_CALCULATED** = 2

*The Jacobian values will be calculated analytically for the nonlinear equations set.*

- INTEGER(INTG), parameter **SOLVER\_NONLINEAR\_JACOBIAN\_FD\_CALCULATED** = 3

*The Jacobian values will be calculated using finite differences for the nonlinear equations set.*

- INTEGER(INTG), parameter **SOLVER\_NO\_OUTPUT** = 0

*No output from the solver routines.*

- INTEGER(INTG), parameter **SOLVER\_TIMING\_OUTPUT** = 1

*Timing output from the solver routines.*

- INTEGER(INTG), parameter **SOLVER\_SOLVER\_OUTPUT** = 2

*Timing and solver specific output from the solver routines.*

- INTEGER(INTG), parameter **SOLVER\_MATRIX\_OUTPUT** = 3

*Timing and solver specific output and solution matrices output from the solver routines.*

- INTEGER(INTG), parameter **SOLVER\_SPARSE\_MATRICES** = 1

*Use sparse solver matrices.*

- INTEGER(INTG), parameter **SOLVER\_FULL\_MATRICES** = 2

*Use fully populated solver matrices.*

- INTEGER(INTG), parameter **SOLVER\_EULER\_INTEGRATOR** = 1

- INTEGER(INTG), parameter **SOLVER\_IMPROVED\_EULER\_INTEGRATOR** = 2

- INTEGER(INTG), parameter **SOLVER\_4TH\_RUNGE\_KUTTA\_INTEGRATOR** = 3

- INTEGER(INTG), parameter **SOLVER\_ADAMS\_MOULTON\_INTEGRATOR** = 4

- INTEGER(INTG), parameter **SOLVER\_LSODA\_INTEGRATOR** = 5

### 6.41.1 Detailed Description

This module handles all solver routines.

### 6.41.2 Function Documentation

#### 6.41.2.1 subroutine **SOLVER\_ROUTINES::SOLVER\_CREATE\_FINISH**

```
(TYPE(SOLVER_TYPE),pointer SOLVER, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)
```

Finishes the process of creating a solver for a problem solution.

#### Parameters:

**SOLVER** A pointer the solver to finish the creation of.

**ERR** The error code

**ERROR** The error string

Definition at line 231 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `SOLVER_EIGENPROBLEM_CREATE_FINISH()`,    `SOLVER_EIGENPROBLEM_TYPE`,    `SOLVER_FINALISE()`,    `SOLVER_LINEAR_CREATE_FINISH()`, `SOLVER_LINEAR_TYPE`, `SOLVER_NONLINEAR_CREATE_FINISH()`, `SOLVER_NONLINEAR_TYPE`, `SOLVER_TIME_INTEGRATION_CREATE_FINISH()`, and `SOLVER_TIME_INTEGRATION_TYPE`.

Referenced by `FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_PROBLEM_SETUP()`, `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP()`, and `LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SETUP()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.2 subroutine SOLVER\_ROUTINES::SOLVER\_CREATE\_START  
(TYPE(SOLUTION\_TYPE),pointer SOLUTION, INTEGER(INTG),intent(in)  
SOLVE\_TYPE, TYPE(SOLVER\_TYPE),pointer SOLVER, INTEGER(INTG),intent(out)  
ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Starts the process of creating a solver for a problem solution.

#### Parameters:

**SOLUTION** A pointer to the solution to create the solver for.

**SOLVE\_TYPE** The type of solver to create

**See also:**

[SOLVER\\_ROUTINES::SolverTypes](#),[SOLVER\\_ROUTINES](#)

**SOLVER** On exit, a pointer to the problem solver.

**ERR** The error code

**ERROR** The error string

Definition at line 280 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `SOLVER_FINALISE()`, and `SOLVER_INITIALISE()`.

Referenced by `FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_PROBLEM_SETUP()`, `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP()`, and `LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SETUP()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.3 subroutine SOLVER\_ROUTINES::SOLVER\_DESTROY (TYPE(SOLVER\_TYPE),pointer SOLVER, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Destroy a problem solver.

#### Parameters:

**SOLVER** A pointer the solver to destroy

**ERR** The error code

**ERROR** The error string

Definition at line 320 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `SOLVER_FINALISE()`.

Referenced by `PROBLEM_ROUTINES::PROBLEM_SOLUTION_FINALISE()`, and `PROBLEM_ROUTINES::PROBLEM_SOLVER_DESTROY()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.4 subroutine `SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_CREATE_FINISH`**  
`(TYPE(EIGENPROBLEM_SOLVER_TYPE),pointer EIGENPROBLEM_SOLVER,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`  
`[private]`

Finishes the process of creating a eigenproblem solver.

#### Parameters:

***EIGENPROBLEM\_SOLVER*** A pointer to the eigenproblem solver to finish the creation of.

**ERR** The error code

**ERROR** The error string

Definition at line 349 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.5 subroutine `SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_FINALISE`**  
`(TYPE(EIGENPROBLEM_SOLVER_TYPE),pointer EIGENPROBLEM_SOLVER,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`  
`[private]`

Finalise a eigenproblem solver for a problem solver.

#### Parameters:

***EIGENPROBLEM\_SOLVER*** A pointer the eigenproblem solver to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 378 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by SOLVER\_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.6 subroutine SOLVER\_ROUTINES::SOLVER\_EIGENPROBLEM\_INITIALISE**  
 (TYPE(SOLVER\_TYPE),pointer *SOLVER*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Initialise a eigenproblem solver for a problem solver.

**Parameters:**

***SOLVER*** A pointer the solver to initialise the eigenproblem solver for  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 405 of file solver\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS(), and SOLVER\_PETSC\_LIBRARY.

Referenced by SOLVER\_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.7 subroutine SOLVER\_ROUTINES::SOLVER\_EIGENPROBLEM\_SOLVE**  
 (TYPE(EIGENPROBLEM\_SOLVER\_TYPE),pointer *EIGENPROBLEM\_SOLVER*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)  
 [private]

Solve a eigenproblem solver.

**Parameters:**

***EIGENPROBLEM\_SOLVER*** A pointer the eigenproblem solver to solve  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 441 of file solver\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Referenced by SOLVER\_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.8 subroutine SOLVER\_ROUTINES::SOLVER\_FINALISE** (TYPE(SOLVER\_-TYPE),pointer *SOLVER*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Finalises a problem solver and deallocates all memory.

**Parameters:**

**SOLVER** A pointer the solver to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 471 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `SOLVER_EIGENPROBLEM_FINALISE()`,    `SOLVER_LINEAR_FINALISE()`,    `SOLVER_NONLINEAR_FINALISE()`, and `SOLVER_TIME_INTEGRATION_FINALISE()`.

Referenced by `SOLVER_CREATE_FINISH()`, `SOLVER_CREATE_START()`, and `SOLVER_DESTROY()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.9 subroutine SOLVER\_ROUTINES::SOLVER\_INITIALISE (TYPE(SOLUTION\_TYPE),pointer SOLUTION, INTEGER(INTG),intent(in) SOLVE\_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Initialise a solver for a problem solution.

**Parameters:**

**SOLUTION** A pointer the solution to initialise the solver for

**SOLVE\_TYPE** The type of solver to create

**See also:**

[SOLVER\\_ROUTINES::SolverTypes](#),[SOLVER\\_ROUTINES](#)

**ERR** The error code

**ERROR** The error string

Definition at line 502 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `SOLVER_EIGENPROBLEM_INITIALISE()`,    `SOLVER_EIGENPROBLEM_TYPE`,    `SOLVER_LINEAR_INITIALISE()`,    `SOLVER_LINEAR_TYPE`,    `SOLVER_NO_OUTPUT`,    `SOLVER_NONLINEAR_INITIALISE()`,    `SOLVER_NONLINEAR_TYPE`,    `SOLVER_SPARSE_MATRICES`, `SOLVER_TIME_INTEGRATION_INITIALISE()`, and `SOLVER_TIME_INTEGRATION_TYPE`.

Referenced by `SOLVER_CREATE_START()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.10 subroutine SOLVER\_ROUTINES::SOLVER\_LIBRARY\_SET (TYPE(SOLVER\_TYPE),pointer SOLVER, INTEGER(INTG),intent(in) SOLVER\_LIBRARY, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets/changes the type of library to use for the solver.

**Parameters:**

**SOLVER** A pointer the solver to set the type of

**SOLVER\_LIBRARY** The type of library to use for the solver

**See also:**

[SOLVER\\_ROUTINES::SolverLibraries](#), [SOLVER\\_ROUTINES](#)

**ERR** The error code

**ERROR** The error string

Definition at line 574 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `SOLVER_CMISS_LIBRARY`, `SOLVER_EIGENPROBLEM_TYPE`, `SOLVER_LINEAR_DIRECT_SOLVE_TYPE`, `SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE`, `SOLVER_LINEAR_TYPE`, `SOLVER_NONLINEAR_LINESEARCH`, `SOLVER_NONLINEAR_TRUSTREGION`, `SOLVER_NONLINEAR_TYPE`, `SOLVER_PETSC_LIBRARY`, and `SOLVER_TIME_INTEGRATION_TYPE`.

Referenced by `FINITE_ELASTICITY_ROUTINES::FINITE_ELASTICITY_PROBLEM_SETUP()`, `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP()`, and `LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SETUP()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.11 subroutine SOLVER\_ROUTINES::SOLVER\_LINEAR\_CREATE\_FINISH  
(TYPE(LINEAR\_SOLVER\_TYPE),pointer LINEAR\_SOLVER,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*) [private]**

Finishes the process of creating a linear solver.

**Parameters:**

**LINEAR\_SOLVER** A pointer to the linear solver to finish the creation of.

**ERR** The error code

**ERROR** The error string

Definition at line 736 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `SOLVER_LINEAR_DIRECT_CREATE_FINISH()`,    `SOLVER_LINEAR_DIRECT_SOLVE_TYPE`,    `SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()`,    and    `SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE`.

Referenced by `SOLVER_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.41.2.12 subroutine SOLVER\_ROUTINES::SOLVER\_LINEAR\_DIRECT\_CREATE\_FINISH**  
 (TYPE(LINEAR\_DIRECT\_SOLVER\_TYPE),pointer *LINEAR\_DIRECT\_SOLVER*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Finishes the process of creating a linear direct solver.

**Parameters:**

***LINEAR\_DIRECT\_SOLVER*** A pointer to the linear direct solver to finish the creation of.

***ERR*** The error code

***ERROR*** The error string

Definition at line 775 of file solver\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(),    SOLVER\_CMISS\_LIBRARY,    SOLVER\_FULL\_MATRICES,    SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRICES\_CREATE\_FINISH(),    SOLVER\_-MATRICES\_ROUTINES::SOLVER\_MATRICES\_CREATE\_START(),    SOLVER\_MATRICES\_-ROUTINES::SOLVER\_MATRICES\_LIBRARY\_TYPE\_SET(),    SOLVER\_MATRICES\_-ROUTINES::SOLVER\_MATRICES\_STORAGE\_TYPE\_SET(),    SOLVER\_PETSC\_LIBRARY,    and    SOLVER\_SPARSE\_MATRICES.

Referenced by SOLVER\_LINEAR\_CREATE\_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.41.2.13 subroutine SOLVER\_ROUTINES::SOLVER\_LINEAR\_DIRECT\_FINALISE**  
 (TYPE(LINEAR\_DIRECT\_SOLVER\_TYPE),pointer *LINEAR\_DIRECT\_SOLVER*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Finalise a direct linear solver for a linear solver and deallocate all memory.

**Parameters:**

***LINEAR\_DIRECT\_SOLVER*** A pointer to the lienar direct solver to finalise

***ERR*** The error code

***ERROR*** The error string

Definition at line 842 of file solver\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Referenced by SOLVER\_LINEAR\_FINALISE(), and SOLVER\_LINEAR\_TYPE\_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.41.2.14 subroutine SOLVER\_ROUTINES::SOLVER\_LINEAR\_DIRECT\_INITIALISE**  
 (TYPE(LINEAR\_SOLVER\_TYPE),pointer *LINEAR\_SOLVER*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Initialise a direct linear solver for a lienar solver.

**Parameters:**

***LINEAR\_SOLVER*** A pointer the linear solver to initialise the direct solver for  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 869 of file solver\_routines.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_-ROUTINES::EXITS(), SOLVER\_CMISS\_LIBRARY, and SOLVER\_DIRECT\_LU.

Referenced by SOLVER\_LINEAR\_TYPE\_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.15 subroutine SOLVER\_ROUTINES::SOLVER\_LINEAR\_DIRECT\_SOLVE**  
 (TYPE(LINEAR\_DIRECT\_SOLVER\_TYPE),pointer *LINEAR\_DIRECT\_SOLVER*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
 \*) [private]

Solve a linear direct solver.

**Parameters:**

***LINEAR\_DIRECT\_SOLVER*** A pointer to the linear direct solver to solve  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 906 of file solver\_routines.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_-ROUTINES::EXITS(), KINDS::PTR, SOLVER\_CMISS\_LIBRARY, and SOLVER\_PETSC\_LIBRARY.

Referenced by SOLVER\_LINEAR\_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.16 subroutine SOLVER\_ROUTINES::SOLVER\_LINEAR\_DIRECT\_TYPE\_SET**  
 (TYPE(SOLVER\_TYPE),pointer *SOLVER*, INTEGER(INTG),intent(in)  
*DIRECT\_SOLVER\_TYPE*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Sets/changes the type of direct linear solver.

**Parameters:**

***SOLVER*** A pointer the problem solver to set the direct linear solver type

**DIRECT\_SOLVER\_TYPE** The type of direct linear solver to set

See also:

[SOLVER\\_ROUTINES::DirectLinearSolverTypes](#), [SOLVER\\_ROUTINES](#)

**ERR** The error code

**ERROR** The error string

Definition at line 987 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `SOLVER_CMISS_LIBRARY`,    `SOLVER_DIRECT_CHOLESKY`,    `SOLVER_DIRECT_LU`,    `SOLVER_DIRECT_SVD`,    `SOLVER_LINEAR_DIRECT_SOLVE_TYPE`,    `SOLVER_LINEAR_TYPE`, and `SOLVER_PETSC_LIBRARY`.

Here is the call graph for this function:

**6.41.2.17 subroutine SOLVER\_ROUTINES::SOLVER\_LINEAR\_FINALISE (TYPE(LINEAR\_SOLVER\_TYPE),pointer LINEAR\_SOLVER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Finalise a linear solver for a problem solver.

**Parameters:**

**LINEAR\_SOLVER** A pointer the linear solver to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 1061 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `SOLVER_LINEAR_DIRECT_FINALISE()`,    and    `SOLVER_LINEAR_ITERATIVE_FINALISE()`.

Referenced by `SOLVER_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.18 subroutine SOLVER\_ROUTINES::SOLVER\_LINEAR\_INITIALISE (TYPE(SOLVER\_TYPE),pointer SOLVER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Initialise a linear solver for a problem solver.

**Parameters:**

**SOLVER** A pointer the solver to initialise the linear solver for

**ERR** The error code

**ERROR** The error string

Definition at line 1090 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `SOLVER_LINEAR_ITERATIVE_INITIALISE()`, and `SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE`.

Referenced by `SOLVER_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.19 subroutine `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_ABSOLUTE_TOLERANCE_SET` (TYPE(SOLVER\_TYPE),pointer *SOLVER*,  
REAL(DP),intent(in) *ABSOLUTE\_TOLERANCE*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Sets/changes the maximum absolute tolerance for an iterative linear solver.

#### Parameters:

***SOLVER*** A pointer the solver to set

***ABSOLUTE\_TOLERANCE*** The absolute tolerance to set

***ERR*** The error code

***ERROR*** The error string

Definition at line 1149 of file `solver_routines.f90`.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE`, and `SOLVER_LINEAR_TYPE`.

Here is the call graph for this function:

**6.41.2.20 subroutine `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH` (TYPE(LINEAR\_ITERATIVE\_SOLVER\_TYPE),pointer  
*LINEAR\_ITERATIVE\_SOLVER*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Finishes the process of creating a linear iterative solver.

#### Parameters:

***LINEAR\_ITERATIVE\_SOLVER*** A pointer to the linear iterative solver to finish the creation of.

***ERR*** The error code

***ERROR*** The error string

Definition at line 1206 of file `solver_routines.f90`.

References    `COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT`,    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `CMISS_PETSC::PETSC_KSPCREATE()`,    `CMISS_PETSC::PETSC_KSPGETPC()`,    `CMISS_PETSC::PETSC_KSPSETFROMOPTIONS()`,    `CMISS_PETSC::PETSC_KSPSETOPERATORS()`,    `CMISS_PETSC::PETSC_KSPSETTOLERANCES()`,    `CMISS_PETSC::PETSC_KSPSETTYPE()`,    `CMISS_PETSC::PETSC_PCSETTYPE()`,    `KINDS::PTR`,    `SOLVER_CMISS_LIBRARY`,    `SOLVER_FULL_MATRICES`,    `SOLVER_ITERATIVE_ADDITIVE_SCHWARZ_PRECONDITIONER`,    `SOLVER_ITERATIVE_BICGSTAB`,    `SOLVER_ITERATIVE_BICONJUGATE_GRADIENT`,    `SOLVER_ITERATIVE_BLOCK_JACOBI_PRECONDITIONER`,    `SOLVER_ITERATIVE_CHEBYCHEV`,

SOLVER\_ITERATIVE\_CONJGRAD\_SQUARED, SOLVER\_ITERATIVE\_CONJUGATE\_GRADIENT, SOLVER\_ITERATIVE\_GMRES, SOLVER\_ITERATIVE\_INCOMPLETE\_CHOLESKY\_PRECONDITIONER, SOLVER\_ITERATIVE\_INCOMPLETE\_LU\_PRECONDITIONER, SOLVER\_ITERATIVE\_JACOBI\_PRECONDITIONER, SOLVER\_ITERATIVE\_NO\_PRECONDITIONER, SOLVER\_ITERATIVE\_RICHARDSON, SOLVER\_ITERATIVE\_SOR\_PRECONDITIONER, SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRICES\_CREATE\_FINISH(), SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRICES\_CREATE\_START(), SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRICES\_LIBRARY\_TYPE\_SET(), SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRICES\_STORAGE\_TYPE\_SET(), SOLVER\_PETSC\_LIBRARY, and SOLVER\_SPARSE\_MATRICES.

Referenced by SOLVER\_LINEAR\_CREATE\_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.21 subroutine SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_DIVERGENCE\_TOLERANCE\_SET (TYPE(SOLVER\_TYPE),pointer SOLVER, REAL(DP),intent(in) DIVERGENCE\_TOLERANCE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Sets/changes the maximum divergence tolerance for an iterative linear solver.

#### Parameters:

**SOLVER** A pointer the problem solver to set

**DIVERGENCE\_TOLERANCE** The divergence tolerance to set

**ERR** The error code

**ERROR** The error string

Definition at line 1345 of file solver\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), SOLVER\_LINEAR\_ITERATIVE\_SOLVE\_TYPE, and SOLVER\_LINEAR\_TYPE.

Here is the call graph for this function:

**6.41.2.22 subroutine SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_FINALISE (TYPE(LINEAR\_ITERATIVE\_SOLVER\_TYPE),pointer LINEAR\_ITERATIVE\_SOLVER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Finalise an iterative linear solver for a linear solver and deallocate all memory.

#### Parameters:

**LINEAR\_ITERATIVE\_SOLVER** A pointer the linear iterative solver to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 1403 of file solver\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `CMISS_PETSC::PETSC_KSPFINALISE()`, and `CMISS_PETSC::PETSC_PCFINALISE()`.

Referenced by `SOLVER_LINEAR_FINALISE()`, and `SOLVER_LINEAR_TYPE_SET()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.23 subroutine SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_INITIALISE  
(TYPE(LINEAR\_SOLVER\_TYPE),pointer LINEAR\_SOLVER,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*) [private]**

Initialise an iterative linear solver for a linear solver.

**Parameters:**

**LINEAR\_SOLVER** A pointer the linear solver to initialise the iterative solver for

**ERR** The error code

**ERROR** The error string

Definition at line 1432 of file solver\_routines.f90.

References `KINDS::_DP`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `CMISS_PETSC::PETSC_KSPINITIALISE()`, `CMISS_PETSC::PETSC_PCNINITIALISE()`, `SOLVER_ITERATIVE_GMRES`, `SOLVER_ITERATIVE_JACOBI_PRECONDITIONER`, and `SOLVER_PETSC_LIBRARY`.

Referenced by `SOLVER_LINEAR_INITIALISE()`, and `SOLVER_LINEAR_TYPE_SET()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.24 subroutine SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_MAXIMUM\_ITERATIONS\_SET  
(TYPE(SOLVER\_TYPE),pointer SOLVER,  
INTEGER(INTG),intent(in) MAXIMUM\_ITERATIONS, INTEGER(INTG),intent(out)  
ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets/changes the maximum number of iterations for an iterative linear solver.

**Parameters:**

**SOLVER** A pointer the problem solver to set the maximum number of iterations

**MAXIMUM\_ITERATIONS** The maximum number of iterations

**ERR** The error code

**ERROR** The error string

Definition at line 1476 of file solver\_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE`, and `SOLVER_LINEAR_TYPE`.

Here is the call graph for this function:

---

**6.41.2.25 subroutine SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_-  
PRECONDITIONER\_TYPE\_SET (TYPE(SOLVER\_TYPE),pointer SOLVER,  
INTEGER(INTG),intent(in) ITERATIVE\_PRECONDITIONER\_TYPE,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*)**

Sets/changes the type of preconditioner for an iterative linear solver.

**Parameters:**

**SOLVER** A pointer the problem solver to set the iterative linear solver type

**ITERATIVE\_PRECONDITIONER\_TYPE** The type of iterative preconditioner to set

**See also:**

[SOLVER\\_ROUTINES::IterativeLinearSolverTypes](#), [SOLVER\\_ROUTINES](#)

**ERR** The error code

**ERROR** The error string

Definition at line 1533 of file solver\_routines.f90.

References    [BASE\\_ROUTINES::ENTERS\(\)](#),    [BASE\\_ROUTINES::ERRORS\(\)](#),    [BASE\\_ROUTINES::EXITS\(\)](#),    [SOLVER\\_ITERATIVE\\_ADDITIVE\\_SCHWARZ\\_PRECONDITIONER](#),    [SOLVER\\_ITERATIVE\\_BLOCK\\_JACOBI\\_PRECONDITIONER](#),    [SOLVER\\_ITERATIVE\\_INCOMPLETE\\_CHOLESKY\\_PRECONDITIONER](#),    [SOLVER\\_ITERATIVE\\_INCOMPLETE\\_LU\\_PRECONDITIONER](#),    [SOLVER\\_ITERATIVE\\_JACOBI\\_PRECONDITIONER](#),    [SOLVER\\_ITERATIVE\\_NO\\_PRECONDITIONER](#),    [SOLVER\\_ITERATIVE\\_SOR\\_PRECONDITIONER](#),    [SOLVER\\_LINEAR\\_ITERATIVE\\_SOLVE\\_TYPE](#),    [SOLVER\\_LINEAR\\_TYPE](#), and [SOLVER\\_PETSC\\_LIBRARY](#).

Here is the call graph for this function:

---

**6.41.2.26 subroutine SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_-  
RELATIVE\_TOLERANCE\_SET (TYPE(SOLVER\_TYPE),pointer SOLVER,  
REAL(DP),intent(in) RELATIVE\_TOLERANCE, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets/changes the relative tolerance for an iterative linear solver.

**Parameters:**

**SOLVER** A pointer the solver to set

**RELATIVE\_TOLERANCE** The relative tolerance to set

**ERR** The error code

**ERROR** The error string

Definition at line 1619 of file solver\_routines.f90.

References    [BASE\\_ROUTINES::ENTERS\(\)](#),    [BASE\\_ROUTINES::ERRORS\(\)](#),    [BASE\\_ROUTINES::EXITS\(\)](#),    [SOLVER\\_LINEAR\\_ITERATIVE\\_SOLVE\\_TYPE](#), and [SOLVER\\_LINEAR\\_TYPE](#).

Here is the call graph for this function:

---

**6.41.2.27 subroutine SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_SOLVE  
(TYPE(LINEAR\_ITERATIVE\_SOLVER\_TYPE),pointer LINEAR\_ITERATIVE\_-  
SOLVER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out)  
ERROR, \*) [private]**

Solves a linear iterative linear solver.

**Parameters:**

**LINEAR\_ITERATIVE\_SOLVER** A pointer the linear iterative solver to solve  
**ERR** The error code  
**ERROR** The error string

Definition at line 1676 of file solver\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(),    BASE\_ROUTINES::GENERAL\_OUTPUT\_TYPE,    CMISS\_PETSC::PETSC\_-KSPGETCONVERGEDREASON(),    CMISS\_PETSC::PETSC\_KSPGETITERATIONNUMBER(),    CMISS\_PETSC::PETSC\_KSPGETRESIDUALNORM(),    CMISS\_PETSC::PETSC\_KSPSOLVE(),    KINDS::PTR,    SOLVER\_CMISS\_LIBRARY,    SOLVER\_PETSC\_LIBRARY, and    SOLVER\_SOLVER\_-OUTPUT.

Referenced by SOLVER\_LINEAR\_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.28 subroutine SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_TYPE\_SET  
(TYPE(SOLVER\_TYPE),pointer SOLVER, INTEGER(INTG),intent(in)  
ITERATIVE\_SOLVER\_TYPE, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Sets/changes the type of iterative linear solver.

**Parameters:**

**SOLVER** A pointer the solver to set the iterative linear solver type  
**ITERATIVE\_SOLVER\_TYPE** The type of iterative linear solver to set

**See also:**

[SOLVER\\_ROUTINES::IterativeLinearSolverTypes](#), [SOLVER\\_ROUTINES](#)

**ERR** The error code

**ERROR** The error string

Definition at line 1820 of file solver\_routines.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(),    SOLVER\_ITERATIVE\_BICGSTAB,    SOLVER\_ITERATIVE\_BICONJUGATE\_-GRADIENT,    SOLVER\_ITERATIVE\_CHEBYCHEV,    SOLVER\_ITERATIVE\_CONJGRAD\_SQUARED,    SOLVER\_ITERATIVE\_CONJUGATE\_GRADIENT,    SOLVER\_ITERATIVE\_GMRES,    SOLVER\_-ITERATIVE\_RICHARDSON,    SOLVER\_LINEAR\_ITERATIVE\_SOLVE\_TYPE,    SOLVER\_LINEAR\_-TYPE, and    SOLVER\_PETSC\_LIBRARY.

Here is the call graph for this function:

---

**6.41.2.29 subroutine SOLVER\_ROUTINES::SOLVER\_LINEAR\_SOLVE (TYPE(LINEAR\_-  
SOLVER\_TYPE),pointer LINEAR\_SOLVER, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Solve a linear solver.

**Parameters:**

**LINEAR\_SOLVER** A pointer to the linear solver to solve  
**ERR** The error code  
**ERROR** The error string

Definition at line 1901 of file solver\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    **BASE\_-ROUTINES::EXITS()**,    **SOLVER\_LINEAR\_DIRECT\_SOLVE()**,    **SOLVER\_LINEAR\_DIRECT\_-SOLVE\_TYPE**, **SOLVER\_LINEAR\_ITERATIVE\_SOLVE()**, and **SOLVER\_LINEAR\_ITERATIVE\_-SOLVE\_TYPE**.

Referenced by **SOLVER\_SOLVE()**.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.30 subroutine SOLVER\_ROUTINES::SOLVER\_LINEAR\_TYPE\_SET (TYPE(SOLVER\_-  
TYPE),pointer SOLVER, INTEGER(INTG),intent(in) LINEAR\_SOLVE\_TYPE,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*)**

Sets/changes the type of linear solver.

**Parameters:**

**SOLVER** A pointer the solver to set the linear solver type

**LINEAR\_SOLVE\_TYPE** The type of linear solver to set

**See also:**

[SOLVER\\_ROUTINES::LinearSolverTypes](#),[SOLVER\\_ROUTINES](#)

**ERR** The error code

**ERROR** The error string

Definition at line 1940 of file solver\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    **BASE\_-ROUTINES::EXITS()**,    **SOLVER\_LINEAR\_DIRECT\_FINALISE()**,    **SOLVER\_LINEAR\_DIRECT\_-INITIALISE()**,    **SOLVER\_LINEAR\_DIRECT\_SOLVE\_TYPE**,    **SOLVER\_LINEAR\_ITERATIVE\_-FINALISE()**,    **SOLVER\_LINEAR\_ITERATIVE\_INITIALISE()**,    **SOLVER\_LINEAR\_ITERATIVE\_-SOLVE\_TYPE**, and **SOLVER\_LINEAR\_TYPE**.

Here is the call graph for this function:

**6.41.2.31 subroutine SOLVER\_ROUTINES::SOLVER\_MATRICES\_ASSEMBLE  
(TYPE(SOLVER\_TYPE),pointer SOLVER, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Assembles the solver matrices and rhs from the equations.

**Parameters:**

**SOLVER** A pointer to the solver

**ERR** The error code

**ERROR** The error string

Definition at line 2017 of file solver\_routines.f90.

References KINDS::\_DP, TIMER::CPU\_TIMER(), BASE\_ROUTINES::ENTERS(), EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_FIXED\_BOUNDARY\_CONDITION, EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_MIXED\_BOUNDARY\_CONDITION, EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_NOT\_FIXED, BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), FIELD\_ROUTINES::FIELD\_VALUES\_SET\_TYPE, BASE\_ROUTINES::GENERAL\_OUTPUT\_TYPE, KINDS::PTR, and SOLVER\_TIMING\_OUTPUT.

Referenced by PROBLEM\_ROUTINES::PROBLEM\_SOLUTION\_JACOBIAN\_EVALUATE(), PROBLEM\_ROUTINES::PROBLEM\_SOLUTION\_RESIDUAL\_EVALUATE(), SOLVER\_NONLINEAR\_RESIDUAL\_EVALUATE(), and SOLVER\_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.32 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_ABSOLUTE\_TOLERANCE\_SET (TYPE(SOLVER\_TYPE),pointer SOLVER, REAL(DP),intent(in) ABSOLUTE\_TOLERANCE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Sets/changes the maximum absolute tolerance for a nonlinear solver.

**Parameters:**

**SOLVER** A pointer the solver to set the absolute tolerance for

**ABSOLUTE\_TOLERANCE** The absolute tolerance to set

**ERR** The error code

**ERROR** The error string

Definition at line 2701 of file solver\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and SOLVER\_NONLINEAR\_TYPE.

Here is the call graph for this function:

**6.41.2.33 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_CREATE\_FINISH (TYPE(NONLINEAR\_SOLVER\_TYPE),pointer NONLINEAR\_SOLVER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Finishes the process of creating a nonlinear solver.

**Parameters:**

**NONLINEAR\_SOLVER** A pointer to the nonlinear solver to finish the creation of.

**ERR** The error code

**ERROR** The error string

Definition at line 2752 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `SOLVER_NONLINEAR_LINESEARCH`,    `SOLVER_NONLINEAR_LINESEARCH_CREATE_FINISH()`,    `SOLVER_NONLINEAR_TRUSTREGION`,    and    `SOLVER_NONLINEAR_TRUSTREGION_CREATE_FINISH()`.

Referenced by `SOLVER_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.34 subroutine `SOLVER_ROUTINES::SOLVER_NONLINEAR_FINALISE`**  
`(TYPE(NONLINEAR_SOLVER_TYPE),pointer NONLINEAR_SOLVER,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`  
`*) [private]`

Finalise a nonlinear solver for a problem solver.

#### Parameters:

***NONLINEAR\_SOLVER*** A pointer the nonlinear solver to finalise

***ERR*** The error code

***ERROR*** The error string

Definition at line 2791 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.35 subroutine `SOLVER_ROUTINES::SOLVER_NONLINEAR_INITIALISE`**  
`(TYPE(SOLVER_TYPE),pointer SOLVER, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Initialise a nonlinear solver for a problem solver.

#### Parameters:

***SOLVER*** A pointer the solver to initialise the nonlinear solver for

***ERR*** The error code

***ERROR*** The error string

Definition at line 2818 of file solver\_routines.f90.

References `KINDS::_DP`,    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `SOLVER_NONLINEAR_LINESEARCH`,    and    `SOLVER_NONLINEAR_LINESEARCH_INITIALISE()`.

Referenced by SOLVER\_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.36 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_JACOBIAN\_EVALUATE  
(TYPE(SOLVER\_TYPE),pointer *SOLVER*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Evaluates the Jacobian for a Newton like nonlinear solver.

**Parameters:**

***SOLVER*** A pointer the solver to evaluate the Jacobian for

***ERR*** The error code

***ERROR*** The error string

Definition at line 2881 of file solver\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by SOLVER\_NONLINEAR\_JACOBIAN\_EVALUATE\_PETSC().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.37 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_-  
JACOBIAN\_EVALUATE\_PETSC (INTEGER(INTG),dimension(\*) *SNES*,  
INTEGER(INTG),dimension(\*) *X*, INTEGER(INTG),dimension(\*) *A*,  
INTEGER(INTG),dimension(\*) *B*, INTEGER(INTG),dimension(\*) *FLAG*,  
TYPE(SOLVER\_TYPE),pointer *CTX*) [private]**

Called from the PETSc SNES solvers to evaluate the Jacobian for a Newton like nonlinear solver.

**Parameters:**

***SNES*** The PETSc SNES type

***X*** The PETSc X Vec type

***A*** The PETSc A Mat type

***B*** The PETSc B Mat type

***FLAG*** The PETSc matrix structure flag

***CTX*** The passed through context

Definition at line 2914 of file solver\_routines.f90.

References SOLVER\_NONLINEAR\_JACOBIAN\_EVALUATE().

Referenced by SOLVER\_NONLINEAR\_LINESEARCH\_CREATE\_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.41.2.38 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_-  
LINESEARCH\_ALPHA\_SET (TYPE(SOLVER\_TYPE),pointer SOLVER,  
REAL(DP),intent(in) LINESEARCH\_ALPHA, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Sets/changes the line search alpha for a nonlinear solver.

**Parameters:**

**SOLVER** A pointer the solver to set the line search alpha for

**LINESEARCH\_ALPHA** The line search alpha to set

**ERR** The error code

**ERROR** The error string

Definition at line 2939 of file solver\_routines.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_-ROUTINES::EXITS(), SOLVER\_NONLINEAR\_LINESEARCH, and SOLVER\_NONLINEAR\_TYPE.

Here is the call graph for this function:

**6.41.2.39 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_-  
CREATE\_FINISH (TYPE(NONLINEAR\_LINESEARCH\_SOLVER\_TYPE),pointer  
NONLINEAR\_LINESEARCH\_SOLVER, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Finishes the process of creating nonlinear Newton line search solver.

**Parameters:**

**NONLINEAR\_LINESEARCH\_SOLVER** A pointer the nonlinear Newton line search solver to finish  
the creation of

**ERR** The error code

**ERROR** The error string

Definition at line 3000 of file solver\_routines.f90.

References      COMP\_ENVIRONMENT::COMPUTATIONAL\_ENVIRONMENT,      BASE\_-ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_ROUTINES::EXITS(),  
CMISS\_PETSC::PETSC\_ISCOLORINGDESTROY(),      CMISS\_PETSC::PETSC\_-MATFDCOLORINGCREATE(), CMISS\_PETSC::PETSC\_MATFDCOLORINGSETFROMOPTIONS(),  
CMISS\_PETSC::PETSC\_MATGETCOLORING(),      CMISS\_PETSC::PETSC\_SNES\_LINESEARCH\_-CUBIC,      CMISS\_PETSC::PETSC\_SNES\_LINESEARCH\_NO,      CMISS\_PETSC::PETSC\_SNES\_-LINESEARCH\_NONORMS,      CMISS\_PETSC::PETSC\_SNES\_LINESEARCH\_QUADRATIC,  
CMISS\_PETSC::PETSC\_SNESCREATE(),      CMISS\_PETSC::PETSC\_SNESLINESEARCHSET(),  
CMISS\_PETSC::PETSC\_SNESLINESEARCHSETPARAMS(),      CMISS\_PETSC::PETSC\_-SNESSETFROMOPTIONS(),      CMISS\_PETSC::PETSC\_SNESSETFUNCTION(),      CMISS\_-PETSC::PETSC\_SNESSETTOLERANCES(),      CMISS\_PETSC::PETSC\_SNESSETTYPE(),  
PROBLEM\_SOLUTION\_RESIDUAL\_EVALUATE\_PETSC(),      KINDS::PTR,      SOLVER\_CMISS\_-LIBRARY,      SOLVER\_FULL\_MATRICES,      SOLVER\_MATRICES\_ROUTINES::SOLVER\_-MATRICES\_CREATE\_FINISH(),      SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRICES\_-CREATE\_START(),      SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRICES\_LIBRARY\_-TYPE\_SET(),      SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRICES\_STORAGE\_TYPE\_-SET(),      SOLVER\_NONLINEAR\_CUBIC\_LINESEARCH,      SOLVER\_NONLINEAR\_JACOBIAN\_-ANALYTIC\_CALCULATED,      SOLVER\_NONLINEAR\_JACOBIAN\_EVALUATE\_PETSC(),

SOLVER\_NONLINEAR\_JACOBIAN\_FD\_CALCULATED, SOLVER\_NONLINEAR\_JACOBIAN\_NOT\_CALCULATED, SOLVER\_NONLINEAR\_NO\_LINESEARCH, SOLVER\_NONLINEAR\_NORMS\_LINESEARCH, SOLVER\_NONLINEAR\_QUADRATIC\_LINESEARCH, SOLVER\_PETSC\_LIBRARY, and SOLVER\_SPARSE\_MATRICES.

Referenced by SOLVER\_NONLINEAR\_CREATE\_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.40 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_FINALISE (TYPE(NONLINEAR\_LINESEARCH\_SOLVER\_TYPE),pointer NONLINEAR\_LINESEARCH\_SOLVER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Finalise a nonlinear Newton line search solver and deallocate all memory.

#### Parameters:

**NONLINEAR\_LINESEARCH\_SOLVER** A pointer the nonlinear Newton line search solver to finalise

**ERR** The error code

**ERROR** The error string

Definition at line 3165 of file solver\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), CMISS\_PETSC::PETSC\_ISCOLORINGFINALISE(), CMISS\_PETSC::PETSC\_MATFDCOLORINGFINALISE(), and CMISS\_PETSC::PETSC\_SNESFINALISE().

Referenced by SOLVER\_NONLINEAR\_TYPE\_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.41 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_INITIALISE (TYPE(NONLINEAR\_SOLVER\_TYPE),pointer NONLINEAR\_SOLVER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Initialise a nonlinear Newton line search solver for a problem solver.

#### Parameters:

**NONLINEAR\_SOLVER** A pointer the nonlinear solver to initialise the Newton line search solver for

**ERR** The error code

**ERROR** The error string

Definition at line 3195 of file solver\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), CMISS\_PETSC::PETSC\_ISCOLORINGINITIALISE(), CMISS\_PETSC::PETSC\_MATFDCOLORINGINITIALISE(), CMISS\_PETSC::PETSC\_SNESINITIALISE(),

SOLVER\_NONLINEAR\_CUBIC\_LINESEARCH,  
CALCULATED, and SOLVER\_PETSC\_LIBRARY.

Referenced by SOLVER\_NONLINEAR\_INITIALISE(), and SOLVER\_NONLINEAR\_TYPE\_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.42 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_-  
MAXSTEP\_SET (TYPE(SOLVER\_TYPE),pointer SOLVER, REAL(DP),intent(in)  
LINESEARCH\_MAXSTEP, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Sets/changes the line search maximum step for a nonlinear solver.

**Parameters:**

**SOLVER** A pointer the solver to set the line search maximum step for

**LINESEARCH\_MAXSTEP** The line search maximum step to set

**ERR** The error code

**ERROR** The error string

Definition at line 3239 of file solver\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_-ROUTINES::EXITS(), SOLVER\_NONLINEAR\_LINESEARCH, and SOLVER\_NONLINEAR\_TYPE.

Here is the call graph for this function:

**6.41.2.43 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_SOLVE  
(TYPE(NONLINEAR\_LINESEARCH\_SOLVER\_TYPE),pointer  
NONLINEAR\_LINESEARCH\_SOLVER, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

**Parameters:**

**NONLINEAR\_LINESEARCH\_SOLVER** A pointer to the nonlinear Newton line search solver to solve

**ERR** The error code

**ERROR** The error string

Definition at line 3301 of file solver\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_-ROUTINES::EXITS(), BASE\_ROUTINES::GENERAL\_OUTPUT\_TYPE, CMISS\_PETSC::PETSC\_SNESGETCONVERGEDREASON(), CMISS\_PETSC::PETSC\_SNESGETFUNCTIONNORM(), CMISS\_PETSC::PETSC\_SNESGETITERATIONNUMBER(), CMISS\_PETSC::PETSC\_SNESSOLVE(), KINDS::PTR, SOLVER\_CMISS\_LIBRARY, SOLVER\_PETSC\_LIBRARY, and SOLVER\_SOLVER\_OUTPUT.

Referenced by SOLVER\_NONLINEAR\_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.41.2.44 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_STEPTOL\_SET (TYPE(SOLVER\_TYPE),pointer SOLVER, REAL(DP),intent(in) LINESEARCH\_STEPTOL, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Sets/changes the line search step tolerance for a nonlinear solver.

**Parameters:**

**SOLVER** A pointer the solver to set the line search step tolerance for  
**LINESEARCH\_STEPTOL** The line search step tolerance to set  
**ERR** The error code  
**ERROR** The error string

Definition at line 3417 of file solver\_routines.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_ROUTINES::EXITS(), SOLVER\_NONLINEAR\_LINESEARCH, and SOLVER\_NONLINEAR\_TYPE.

Here is the call graph for this function:

**6.41.2.45 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH\_TYPE\_SET (TYPE(SOLVER\_TYPE),pointer SOLVER, INTEGER(INTG),intent(in) LINESEARCH\_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Sets/changes the line search type for a nonlinear solver.

**Parameters:**

**SOLVER** A pointer the solver to set the line search type for  
**LINESEARCH\_TYPE** The line search type to set

**See also:**

**SOLVER\_ROUTINES::NonlinearLineSearchTypes,SOLVER\_ROUTINES**  
**ERR** The error code  
**ERROR** The error string

Definition at line 3479 of file solver\_routines.f90.

References     BASE\_ROUTINES::ENTERS(),     BASE\_ROUTINES::ERRORS(),     BASE\_ROUTINES::EXITS(), SOLVER\_NONLINEAR\_CUBIC\_LINESEARCH, SOLVER\_NONLINEAR\_LINESEARCH, SOLVER\_NONLINEAR\_NO\_LINESEARCH, SOLVER\_NONLINEAR\_NONORMS\_LINESEARCH, SOLVER\_NONLINEAR\_QUADRATIC\_LINESEARCH, and SOLVER\_NONLINEAR\_TYPE.

Here is the call graph for this function:

**6.41.2.46 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_MAXIMUM\_FUNCTION\_EVALUATIONS\_SET (TYPE(SOLVER\_TYPE),pointer SOLVER, INTEGER(INTG),intent(in) MAXIMUM\_FUNCTION\_EVALUATIONS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Sets/changes the maximum number of function evaluations for a nonlinear solver.

**Parameters:**

**SOLVER** A pointer the solver to set the maximum function evaluations for  
**MAXIMUM\_FUNCTION\_EVALUATIONS** The maximum function evaluations to set  
**ERR** The error code  
**ERROR** The error string

Definition at line 3547 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `SOLVER_NONLINEAR_TYPE`.

Here is the call graph for this function:

**6.41.2.47 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_MAXIMUM\_ITERATIONS\_SET (TYPE(SOLVER\_TYPE),pointer SOLVER, INTEGER(INTG),intent(in) MAXIMUM\_ITERATIONS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Sets/changes the maximum number of iterations for a nonlinear solver.

**Parameters:**

**SOLVER** A pointer the solver to set the maximum iterations for  
**MAXIMUM\_ITERATIONS** The maximum iterations to set  
**ERR** The error code  
**ERROR** The error string

Definition at line 3599 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `SOLVER_NONLINEAR_TYPE`.

Here is the call graph for this function:

**6.41.2.48 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_RELATIVE\_TOLERANCE\_SET (TYPE(SOLVER\_TYPE),pointer SOLVER, REAL(DP),intent(in) RELATIVE\_TOLERANCE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Sets/changes the relative tolerance for a nonlinear solver.

**Parameters:**

**SOLVER** A pointer the solver to set the relative tolerance for  
**RELATIVE\_TOLERANCE** The relative tolerance to set  
**ERR** The error code  
**ERROR** The error string

Definition at line 3650 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `SOLVER_NONLINEAR_TYPE`.

Here is the call graph for this function:

---

**6.41.2.49 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_RESIDUAL\_EVALUATE**  
**(TYPE(SOLVER\_TYPE),pointer SOLVER, INTEGER(INTG),intent(out) ERR,**  
**TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Evaluates the residual for a Newton like nonlinear solver.

**Parameters:**

**SOLVER** A pointer the solver to evaluate the residual for

**ERR** The error code

**ERROR** The error string

Definition at line 3701 of file solver\_routines.f90.

References    **BASE\_ROUTINES::ENTERS()**,    **BASE\_ROUTINES::ERRORS()**,    **BASE\_-ROUTINES::EXITS()**,    **KINDS::PTR**,    **SOLVER\_MATRICES\_ASSEMBLE()**,    and    **SOLVER\_-VARIABLES\_UPDATE()**.

Referenced by **SOLVER\_NONLINEAR\_RESIDUAL\_EVALUATE\_PETSC()**.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.50 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_-  
RESIDUAL\_EVALUATE\_PETSC (INTEGER(INTG),dimension(\*) SNES,  
INTEGER(INTG),dimension(\*) X, INTEGER(INTG),dimension(\*) F,  
TYPE(SOLVER\_TYPE),pointer CTX) [private]**

Called from the PETSc SNES solvers to evaluate the residual for a Newton like nonlinear solver.

**Parameters:**

**SNES** The PETSc SNES type

**X** The PETSc X Vec type

**F** The PETSc F Vec type

**CTX** The passed through context

Definition at line 3756 of file solver\_routines.f90.

References **SOLVER\_NONLINEAR\_RESIDUAL\_EVALUATE()**.

Here is the call graph for this function:

**6.41.2.51 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_SOLUTION\_-  
TOLERANCE\_SET (TYPE(SOLVER\_TYPE),pointer SOLVER,  
REAL(DP),intent(in) SOLUTION\_TOLERANCE, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Sets/changes the solution tolerance for a nonlinear solver.

**Parameters:**

**SOLVER** A pointer the solver to set the solution tolerance for

**SOLUTION\_TOLERANCE** The solution tolerance to set

**ERR** The error code

**ERROR** The error string

Definition at line 3779 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `SOLVER_NONLINEAR_TYPE`.

Here is the call graph for this function:

```
6.41.2.52 subroutine SOLVER_ROUTINES::SOLVER_NONLINEAR_SOLVE
 (TYPE(NONLINEAR_SOLVER_TYPE),pointer NONLINEAR_SOLVER,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *) [private]
```

#### Parameters:

**NONLINEAR\_SOLVER** A pointer to the nonlinear solver to solve

**ERR** The error code

**ERROR** The error string

Definition at line 3830 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `SOLVER_NONLINEAR_LINESEARCH`,    `SOLVER_NONLINEAR_LINESEARCH_SOLVE()`, `SOLVER_NONLINEAR_TRUSTREGION`, and `SOLVER_NONLINEAR_TRUSTREGION_SOLVE()`.

Referenced by `SOLVER_SOLVE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

```
6.41.2.53 subroutine SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_
 CREATE_FINISH (TYPE(NONLINEAR_TRUSTREGION_SOLVER_TYPE),pointer
 NONLINEAR_TRUSTREGION_SOLVER, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Finishes the process of creating nonlinear trust region solver.

#### Parameters:

**NONLINEAR\_TRUSTREGION\_SOLVER** A pointer the nonlinear trust region solver to finish the creation of

**ERR** The error code

**ERROR** The error string

Definition at line 3869 of file solver\_routines.f90.

References    `COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT`,    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `CMISS_PETSC::PETSC_SNESCREATE()`, `CMISS_PETSC::PETSC_SNESSETFROMOPTIONS()`, `CMISS_PETSC::PETSC_SNESSETFUNCTION()`,    `CMISS_PETSC::PETSC_SNESSETTOLERANCES()`,

CMISS\_PETSC::PETSC\_SNESSETTRUSTREGIONTOLERANCE(), CMISS\_PETSC::PETSC\_SNESSETTYPE(), PROBLEM\_SOLUTION\_RESIDUAL\_EVALUATE\_PETSC(), SOLVER\_CMISS\_LIBRARY, SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRICES\_CREATE\_FINISH(), SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRICES\_CREATE\_START(), SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRICES\_LIBRARY\_TYPE\_SET(), and SOLVER\_PETSC\_LIBRARY.

Referenced by SOLVER\_NONLINEAR\_CREATE\_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.54 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_TRUSTREGION\_DELTA0\_SET (TYPE(SOLVER\_TYPE),pointer SOLVER, REAL(DP),intent(in) TRUSTREGION\_DELTA0, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Sets/changes the trust region delta0 for a nonlinear solver.

#### Parameters:

**SOLVER** A pointer the solver to set the trust region delta0 for  
**TRUSTREGION\_DELTA0** The trust region delta0 to set  
**ERR** The error code  
**ERROR** The error string

Definition at line 3962 of file solver\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), SOLVER\_NONLINEAR\_TRUSTREGION, and SOLVER\_NONLINEAR\_TYPE.

Here is the call graph for this function:

**6.41.2.55 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_TRUSTREGION\_FINALISE (TYPE(NONLINEAR\_TRUSTREGION\_SOLVER\_TYPE),pointer NONLINEAR\_TRUSTREGION\_SOLVER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Finalise a nonlinear trust region solver and deallocate all memory.

#### Parameters:

**NONLINEAR\_TRUSTREGION\_SOLVER** A pointer the non linear trust region solver to finalise  
**ERR** The error code  
**ERROR** The error string

Definition at line 4024 of file solver\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), and CMISS\_PETSC::PETSC\_SNESFINALISE().

Referenced by SOLVER\_NONLINEAR\_TYPE\_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.41.2.56 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_TRUSTREGION\_INITIALISE** (TYPE(NONLINEAR\_SOLVER\_TYPE),pointer *NONLINEAR\_SOLVER*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Initialise a nonlinear trust region solver for a problem solver.

**Parameters:**

***NONLINEAR\_SOLVER*** A pointer the nonsolver to initialise the trust region solver for  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 4052 of file solver\_routines.f90.

References KINDS::\_DP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), CMISS\_PETSC::PETSC\_SNESINITIALISE(), and SOLVER\_PETSC\_LIBRARY.

Referenced by SOLVER\_NONLINEAR\_TYPE\_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.57 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_TRUSTREGION\_SOLVE** (TYPE(NONLINEAR\_TRUSTREGION\_SOLVER\_TYPE),pointer *NONLINEAR\_TRUSTREGION\_SOLVER*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

**Parameters:**

***NONLINEAR\_TRUSTREGION\_SOLVER*** A pointer to the nonlinear Newton trust region solver to solve  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 4091 of file solver\_routines.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), BASE\_ROUTINES::EXITS(), SOLVER\_CMISS\_LIBRARY, and SOLVER\_PETSC\_LIBRARY.

Referenced by SOLVER\_NONLINEAR\_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.58 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_TRUSTREGION\_TOLERANCE\_SET** (TYPE(SOLVER\_TYPE),pointer *SOLVER*, REAL(DP),intent(in) *TRUSTREGION\_TOLERANCE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Sets/changes the trust region tolerance for a nonlinear solver.

**Parameters:**

***SOLVER*** A pointer the solver to set the trust region tolerance for

**TRUSTREGION\_TOLERANCE** The trust region tolerance to set

**ERR** The error code

**ERROR** The error string

Definition at line 4148 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `SOLVER_NONLINEAR_TRUSTREGION`, and `SOLVER_NONLINEAR_TYPE`.

Here is the call graph for this function:

**6.41.2.59 subroutine SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_TYPE\_SET**  
`(TYPE(SOLVER_TYPE),pointer SOLVER, INTEGER(INTG),intent(in) NONLINEAR_SOLVE_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)` [private]

Sets/changes the type of nonlinear solver.

**Parameters:**

**SOLVER** A pointer the solver to set the nonlinear solver type

**NONLINEAR\_SOLVE\_TYPE** The type of nonlinear solver to set

**See also:**

[SOLVER\\_ROUTINES::NonlinearSolverTypes](#), [SOLVER\\_ROUTINES](#)

**ERR** The error code

**ERROR** The error string

Definition at line 4210 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`,    `SOLVER_NONLINEAR_LINESearch`,    `SOLVER_NONLINEAR_LINESearch_FINALISE()`,    `SOLVER_NONLINEAR_LINESearch_INITIALISE()`,    `SOLVER_NONLINEAR_TRUSTREGION`, `SOLVER_NONLINEAR_TRUSTREGION_FINALISE()`, `SOLVER_NONLINEAR_TRUSTREGION_INITIALISE()`, and `SOLVER_NONLINEAR_TYPE`.

Here is the call graph for this function:

**6.41.2.60 subroutine SOLVER\_ROUTINES::SOLVER\_OUTPUT\_TYPE\_SET**  
`(TYPE(SOLVER_TYPE),pointer SOLVER, INTEGER(INTG),intent(in) OUTPUT_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets/changes the output type for a solver.

**Parameters:**

**SOLVER** A pointer the problem solver to set the iterative linear solver type

**OUTPUT\_TYPE** The type of solver output to be set

**See also:**

[SOLVER\\_ROUTINES::OutputTypes](#), [SOLVER\\_ROUTINES](#)

**ERR** The error code

**ERROR** The error string

Definition at line 4290 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `SOLVER_MATRIX_OUTPUT`, `SOLVER_NO_OUTPUT`, `SOLVER_SOLVER_OUTPUT`, and `SOLVER_TIMING_OUTPUT`.

Here is the call graph for this function:

**6.41.2.61 subroutine `SOLVER_ROUTINES::SOLVER_SOLVE` (`TYPE(SOLVER_TYPE),pointer SOLVER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, */`)**

Solve the problem.

**Parameters:**

**SOLVER** A pointer the solver to solve

**ERR** The error code

**ERROR** The error string

Definition at line 4524 of file solver\_routines.f90.

References `TIMER::CPU_TIMER()`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`,    `BASE_ROUTINES::GENERAL_OUTPUT_TYPE`,    `SOLVER_EIGENPROBLEM_SOLVE()`,    `SOLVER_EIGENPROBLEM_TYPE`,    `SOLVER_LINEAR_SOLVE()`, `SOLVER_LINEAR_TYPE`,    `SOLVER_MATRICES_ASSEMBLE()`,    `SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_OUTPUT()`,    `SOLVER_MATRIX_OUTPUT`,    `SOLVER_NONLINEAR_SOLVE()`,    `SOLVER_NONLINEAR_TYPE`,    `SOLVER_TIME_INTEGRATION_SOLVE()`, `SOLVER_TIME_INTEGRATION_TYPE`, and `SOLVER_TIMING_OUTPUT`.

Referenced by `PROBLEM_ROUTINES::PROBLEM SOLUTION_SOLVE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.62 subroutine `SOLVER_ROUTINES::SOLVER_SPARSITY_TYPE_SET` (`TYPE(SOLVER_TYPE),pointer SOLVER, INTEGER(INTG),intent(in) SPARSITY_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, */`)**

Sets/changes the sparsity type for a solver.

**Parameters:**

**SOLVER** A pointer the problem solver to set the iterative linear solver type

**SPARSITY\_TYPE** The type of solver sparsity to be set

**See also:**

[SOLVER\\_ROUTINES::SparsityTypes](#),[SOLVER\\_ROUTINES](#)

**ERR** The error code

**ERROR** The error string

Definition at line 4338 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `SOLVER_FULL_MATRICES`, and `SOLVER_SPARSE_MATRICES`.

Referenced by    `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP()`,    and    `LINEAR_ELASTICITY_ROUTINES::LINEAR_ELASTICITY_PROBLEM_SETUP()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.63 subroutine SOLVER\_ROUTINES::SOLVER\_TIME\_INTEGRATION\_CREATE\_FINISH (TYPE(TIME\_INTEGRATION\_SOLVER\_TYPE),pointer TIME\_INTEGRATION\_SOLVER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Finishes the process of creating a time integration solver.

#### Parameters:

**TIME\_INTEGRATION\_SOLVER** A pointer to the time integration solver to finish the creation of.  
**ERR** The error code  
**ERROR** The error string

Definition at line 4383 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.64 subroutine SOLVER\_ROUTINES::SOLVER\_TIME\_INTEGRATION\_FINALISE (TYPE(TIME\_INTEGRATION\_SOLVER\_TYPE),pointer TIME\_INTEGRATION\_SOLVER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Finalise a time integration solver for a problem solver.

#### Parameters:

**TIME\_INTEGRATION\_SOLVER** A pointer the time integration solver to finalise  
**ERR** The error code  
**ERROR** The error string

Definition at line 4412 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.41.2.65 subroutine SOLVER\_ROUTINES::SOLVER\_TIME\_INTEGRATION\_INITIALISE**  
`(TYPE(SOLVER_TYPE),pointer SOLVER, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Initialise a time integration solver for a problem solver.

**Parameters:**

***SOLVER*** A pointer the solver to initialise the time integration solver for  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 4439 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_-ROUTINES::EXITS()`, and `SOLVER_PETSC_LIBRARY`.

Referenced by `SOLVER_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.66 subroutine SOLVER\_ROUTINES::SOLVER\_TIME\_INTEGRATION\_SOLVE**  
`(TYPE(TIME_INTEGRATION_SOLVER_TYPE),pointer TIME_INTEGRATION_-`  
`SOLVER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)`  
`ERROR, *) [private]`

Solve a time integration solver.

**Parameters:**

***TIME\_INTEGRATION\_SOLVER*** A pointer to the time integration solver to solve  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 4495 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_-ROUTINES::EXITS()`.

Referenced by `SOLVER_SOLVE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.41.2.67 subroutine SOLVER\_ROUTINES::SOLVER\_VARIABLES\_UPDATE**  
`(TYPE(SOLVER_TYPE),pointer SOLVER, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Updates the dependent variables from the solver solution.

**Parameters:**

***SOLVER*** A pointer the solver to update the variables from

**ERR** The error code

**ERROR** The error string

Definition at line 4602 of file solver\_routines.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `FIELD_ROUTINES::FIELD_VALUES_SET_TYPE`, and `KINDS::PTR`.

Referenced by    `PROBLEM_ROUTINES::PROBLEM_SOLUTION_JACOBIAN_EVALUATE()`,  
`PROBLEM_ROUTINES::PROBLEM_SOLUTION_RESIDUAL_EVALUATE()`,    `PROBLEM_ROUTINES::PROBLEM_SOLUTION_SOLVE()`,    and    `SOLVER_NONLINEAR_RESIDUAL_EVALUATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

### 6.41.3 Variable Documentation

#### 6.41.3.1 INTEGER(INTG),parameter **SOLVER\_ROUTINES::SOLVER\_4TH\_RUNGE\_KUTTA\_INTEGRATOR = 3**

Definition at line 176 of file solver\_routines.f90.

#### 6.41.3.2 INTEGER(INTG),parameter **SOLVER\_ROUTINES::SOLVER\_ADAMS\_MOULTON\_INTEGRATOR = 4**

Definition at line 177 of file solver\_routines.f90.

#### 6.41.3.3 INTEGER(INTG),parameter **SOLVER\_ROUTINES::SOLVER\_EULER\_INTEGRATOR = 1**

Definition at line 174 of file solver\_routines.f90.

#### 6.41.3.4 INTEGER(INTG),parameter **SOLVER\_ROUTINES::SOLVER\_IMPROVED\_EULER\_INTEGRATOR = 2**

Definition at line 175 of file solver\_routines.f90.

#### 6.41.3.5 INTEGER(INTG),parameter **SOLVER\_ROUTINES::SOLVER\_LSODA\_INTEGRATOR = 5**

Definition at line 178 of file solver\_routines.f90.

## 6.42 SORTING Namespace Reference

This module contains all procedures for sorting. NOTE: THE ROUTINES IN THIS MODULE HAVE NOT BEEN TESTED!!!

### Classes

- interface [BUBBLE\\_SORT](#)
- interface [HEAP\\_SORT](#)
- interface [SHELL\\_SORT](#)

### Functions

- subroutine [BUBBLE\\_SORT\\_INTG](#) (A, ERR, ERROR,\*)
- subroutine [BUBBLE\\_SORT\\_SP](#) (A, ERR, ERROR,\*)
- subroutine [BUBBLE\\_SORT\\_DP](#) (A, ERR, ERROR,\*)
- subroutine [HEAP\\_SORT\\_INTG](#) (A, ERR, ERROR,\*)
- subroutine [HEAP\\_SORT\\_SP](#) (A, ERR, ERROR,\*)
- subroutine [HEAP\\_SORT\\_DP](#) (A, ERR, ERROR,\*)
- subroutine [SHELL\\_SORT\\_INTG](#) (A, ERR, ERROR,\*)
- subroutine [SHELL\\_SORT\\_SP](#) (A, ERR, ERROR,\*)
- subroutine [SHELL\\_SORT\\_DP](#) (A, ERR, ERROR,\*)

### 6.42.1 Detailed Description

This module contains all procedures for sorting. NOTE: THE ROUTINES IN THIS MODULE HAVE NOT BEEN TESTED!!!

### 6.42.2 Function Documentation

#### 6.42.2.1 subroutine [SORTING::BUBBLE\\_SORT\\_DP](#) (*REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \**) [private]

Definition at line 185 of file sorting.f90.

References [BASE\\_ROUTINES::ENTERS\(\)](#), [BASE\\_ROUTINES::ERRORS\(\)](#), and [BASE\\_ROUTINES::EXITS\(\)](#).

Here is the call graph for this function:

#### 6.42.2.2 subroutine [SORTING::BUBBLE\\_SORT\\_INTG](#) (*INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*/*) [private]

Definition at line 96 of file sorting.f90.

References [BASE\\_ROUTINES::ENTERS\(\)](#), [BASE\\_ROUTINES::ERRORS\(\)](#), and [BASE\\_ROUTINES::EXITS\(\)](#).

Here is the call graph for this function:

---

**6.42.2.3 subroutine SORTING::BUBBLE\_SORT\_SP (REAL(SP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 140 of file sorting.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.42.2.4 subroutine SORTING::HEAP\_SORT\_DP (REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 362 of file sorting.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.42.2.5 subroutine SORTING::HEAP\_SORT\_INTG (INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 239 of file sorting.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.42.2.6 subroutine SORTING::HEAP\_SORT\_SP (REAL(SP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 300 of file sorting.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.42.2.7 subroutine SORTING::SHELL\_SORT\_DP (REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 524 of file sorting.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.42.2.8 subroutine `SORTING::SHELL_SORT_INTG` (`INTEGER(INTG),dimension(:),intent(inout) A`, `INTEGER(INTG),intent(out) ERR`,  
`TYPE(VARYING_STRING),intent(out) ERROR`, `*`) [private]**

Definition at line 433 of file sorting.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.42.2.9 subroutine `SORTING::SHELL_SORT_SP` (`REAL(SP),dimension(:),intent(inout) A`,  
`INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR`, `*`)  
[private]**

Definition at line 478 of file sorting.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

## 6.43 STRINGS Namespace Reference

This module contains all string manipulation and transformation routines.

### Classes

- interface [IS\\_ABBREVIATION](#)

*Returns .TRUE. if a supplied string is a valid abbreviation of a second supplied string.*

- interface [LIST\\_TO\\_CHARACTER](#)

*Converts a list to its equivalent character string representation.*

- interface [NUMBER\\_TO\\_CHARACTER](#)

*Converts a number to its equivalent character string representation.*

- interface [NUMBER\\_TO\\_VSTRING](#)

*Converts a number to its equivalent varying string representation.*

- interface [STRING\\_TO\\_DOUBLE](#)

*Converts a string representation of a number to a double precision number.*

- interface [STRING\\_TO\\_INTEGER](#)

*Converts a string representation of a number to an integer.*

- interface [STRING\\_TO\\_LONG\\_INTEGER](#)

*Converts a string representation of a number to a long integer.*

- interface [STRING\\_TO\\_LOGICAL](#)

*Converts a string representation of a boolean value (TRUE or FALSE) to a logical.*

- interface [STRING\\_TO\\_SINGLE](#)

*Converts a string representation of a number to a single precision number.*

- interface [CHARACTER\\_TO\\_LOWERCASE](#)

*Returns a character string which is the lowercase equivalent of the supplied string.*

- interface [VSTRING\\_TO\\_LOWERCASE](#)

*Returns a varying string which is the lowercase equivalent of the supplied string.*

- interface [CHARACTER\\_TO\\_UPPERCASE](#)

*Returns a character string which is the uppercase equivalent of the supplied string.*

- interface [VSTRING\\_TO\\_UPPERCASE](#)

*Returns a varying string which is the uppercase equivalent of the supplied string.*

## Functions

- LOGICAL [IS\\_ABBREVIATION\\_C\\_C](#) (SHORT, LONG, MIN\_NUM\_CHARACTERS)  
*IS\_ABBREVIATION* returns .TRUE. if the character string *SHORT* is an abbreviation of the character string *LONG*. *SHORT* must be at least *MIN\_NUM\_CHARACTERS* long.
- LOGICAL [IS\\_ABBREVIATION\\_C\\_VS](#) (SHORT, LONG, MIN\_NUM\_CHARACTERS)  
*IS\_ABBREVIATION* returns .TRUE. if the character string *SHORT* is an abbreviation of the varying string *LONG*. *SHORT* must be at least *MIN\_NUM\_CHARACTERS* long.
- LOGICAL [IS\\_ABBREVIATION\\_VS\\_C](#) (SHORT, LONG, MIN\_NUM\_CHARACTERS)  
*IS\_ABBREVIATION* returns .TRUE. if the varying string *SHORT* is an abbreviation of the character string *LONG*. *SHORT* must be at least *MIN\_NUM\_CHARACTERS* long.
- LOGICAL [IS\\_ABBREVIATION\\_VS\\_VS](#) (SHORT, LONG, MIN\_NUM\_CHARACTERS)  
*IS\_ABBREVIATION* returns .TRUE. if the varying string *SHORT* is an abbreviation of the varying string *LONG*. *SHORT* must be at least *MIN\_NUM\_CHARACTERS* long.
- LOGICAL [IS\\_DIGIT](#) (CHARAC)  
*IS\_DIGIT* returns .TRUE. if the character *CHARAC* is a digit character (i.e. 0..9).
- LOGICAL [IS\\_LETTER](#) (CHARAC)  
*IS\_LETTER* returns .TRUE. if the character *CHARAC* is a letter character (i.e. A..Z or a..z).
- LOGICAL [IS\\_LOWERCASE](#) (CHARC)  
*Returns* .TRUE. *if the supplied character is a lowercase character.*
- LOGICAL [IS\\_UPPERCASE](#) (CHARC)  
*Returns* .TRUE. *if the supplied character is an uppercase character.*
- LOGICAL [IS\\_WHITESPACE](#) (CHARAC)  
*IS\_WHITESPACE* returns .TRUE. if the character *CHARAC* is a whitespace character (i.e. space, tabs, etc.).
- CHARACTER(LEN=MAXSTRLEN) [LIST\\_TO\\_CHARACTER\\_C](#) (NUMBER\_IN\_LIST, LIST, FORMAT, ERR, ERROR, LIST\_LENGTHS)  
*Converts a character list to its equivalent character string representation as determined by the supplied format. If present, the optional argument LIST\_LENGTHS is used for the lengths of each list elements length otherwise the trimmed length is used. NOTE: The FORMAT is ignored for this child FUNCTION.*
- CHARACTER(LEN=MAXSTRLEN) [LIST\\_TO\\_CHARACTER\\_INTG](#) (NUMBER\_IN\_LIST, LIST, FORMAT, ERR, ERROR)  
*Converts an integer list to its equivalent character string representation as determined by the supplied format.*
- CHARACTER(LEN=MAXSTRLEN) [LIST\\_TO\\_CHARACTER\\_LINTG](#) (NUMBER\_IN\_LIST, LIST, FORMAT, ERR, ERROR)  
*Converts an long integer list to its equivalent character string representation as determined by the supplied format.*
- CHARACTER(LEN=MAXSTRLEN) [LIST\\_TO\\_CHARACTER\\_L](#) (NUMBER\_IN\_LIST, LIST, FORMAT, ERR, ERROR)

*Converts a logical list to its equivalent character string representation as determined by the supplied format string. The FORMAT is ignored for this child FUNCTION.*

- CHARACTER(LEN=MAXSTRLEN) [LIST\\_TO\\_CHARACTER\\_SP](#) (NUMBER\_IN\_LIST, LIST, FORMAT, ERR, ERROR)

*Converts a single precision list to its equivalent character string representation as determined by the supplied format string.*

- CHARACTER(LEN=MAXSTRLEN) [LIST\\_TO\\_CHARACTER\\_DP](#) (NUMBER\_IN\_LIST, LIST, FORMAT, ERR, ERROR)

*Converts a double precision list to its equivalent character string representation as determined by the supplied format string.*

- CHARACTER(LEN=MAXSTRLEN) [LOGICAL\\_TO\\_CHARACTER](#) (LOGICALVALUE, ERR, ERROR)

*Converts a logical value to either a "TRUE" or "FALSE" character string.*

- TYPE([VARYING\\_STRING](#)) [LOGICAL\\_TO\\_VSTRING](#) (LOGICALVALUE, ERR, ERROR)

*Converts a logical value to either a "TRUE" or "FALSE" varying string.*

- CHARACTER(LEN=MAXSTRLEN) [NUMBER\\_TO\\_CHARACTER\\_INTG](#) (NUMBER, FORMAT, ERR, ERROR)

*Converts an integer number to its equivalent character string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g. "I3".*

- CHARACTER(LEN=MAXSTRLEN) [NUMBER\\_TO\\_CHARACTER\\_LINTG](#) (NUMBER, FORMAT, ERR, ERROR)

*Converts a long integer number to its equivalent character string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g. "I3".*

- CHARACTER(LEN=MAXSTRLEN) [NUMBER\\_TO\\_CHARACTER\\_SP](#) (NUMBER, FORMAT, ERR, ERROR)

*Converts a single precision number to its equivalent character string representation as determined by the supplied format string. NOTE: If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="\*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.*

- CHARACTER(LEN=MAXSTRLEN) [NUMBER\\_TO\\_CHARACTER\\_DP](#) (NUMBER, FORMAT, ERR, ERROR)

*Converts a double precision number to its equivalent character string representation as determined by the supplied format string. Note If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="\*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.*

- TYPE([VARYING\\_STRING](#)) [NUMBER\\_TO\\_VSTRING\\_INTG](#) (NUMBER, FORMAT, ERR, ERROR)

*Converts an integer number to its equivalent varying string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g. "I3".*

- TYPE([VARYING\\_STRING](#)) [NUMBER\\_TO\\_VSTRING\\_LINTG](#) (NUMBER, FORMAT, ERR, ERROR)

*Converts a long integer number to its equivalent varying string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g., "I3".*

- **TYPE(VARYING\_STRING) NUMBER\_TO\_VSTRING\_SP** (NUMBER, FORMAT, ERR, ERROR)

*Converts a single precision number to its equivalent varying string representation as determined by the supplied format string. Note If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="\*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.*

- **TYPE(VARYING\_STRING) NUMBER\_TO\_VSTRING\_DP** (NUMBER, FORMAT, ERR, ERROR)

*Converts a double precision number to its equivalent varying string representation as determined by the supplied format string. Note If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="\*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.*

- **REAL(DP) STRING\_TO\_DOUBLE\_C** (STRING, ERR, ERROR)

*Converts a character string representation of a number to a double precision number.*

- **REAL(DP) STRING\_TO\_DOUBLE\_VS** (STRING, ERR, ERROR)

*Converts a varying string representation of a number to a double precision number.*

- **INTEGER(INTG) STRING\_TO\_INTEGER\_C** (STRING, ERR, ERROR)

*Converts a character string representation of a number to an integer.*

- **INTEGER(INTG) STRING\_TO\_INTEGER\_VS** (STRING, ERR, ERROR)

*Converts a varying string representation of a number to an integer.*

- **INTEGER(LINTG) STRING\_TO\_LONG\_INTEGER\_C** (STRING, ERR, ERROR)

*Converts a character string representation of a number to a long integer.*

- **INTEGER(LINTG) STRING\_TO\_LONG\_INTEGER\_VS** (STRING, ERR, ERROR)

*Converts a varying string representation of a number to a long integer.*

- **LOGICAL STRING\_TO\_LOGICAL\_C** (STRING, ERR, ERROR)

*Converts a character string representation of a boolean (TRUE or FALSE) to a logical.*

- **LOGICAL STRING\_TO\_LOGICAL\_VS** (STRING, ERR, ERROR)

*Converts a varying string representation of a boolean (TRUE or FALSE) to a logical.*

- **REAL(SP) STRING\_TO\_SINGLE\_C** (STRING, ERR, ERROR)

*Converts a character string representation of a number to a single precision number.*

- **REAL(SP) STRING\_TO\_SINGLE\_VS** (STRING, ERR, ERROR)

*Converts a varying string representation of a number to a single precision number.*

- function **CHARACTER\_TO\_LOWERCASE\_C** (STRING)

- function **CHARACTER\_TO\_LOWERCASE\_VS** (STRING)

*Returns a character string that is the lowercase equivalent of the supplied varying string.*

- **TYPE(VARYING\_STRING) VSTRING\_TO\_LOWERCASE\_C** (STRING)

*Returns a varying string that is the lowercase equivalent of the supplied character string.*

- TYPE(VARYING\_STRING) VSTRING\_TO\_LOWERCASE\_VS (STRING)  
*Returns a varying string that is the lowercase equivalent of the supplied varying string.*
- function CHARACTER\_TO\_UPPERCASE\_C (STRING)  
*Returns a character string which is uppercase equivalent of the supplied character string.*
- function CHARACTER\_TO\_UPPERCASE\_VS (STRING)  
*Returns a character string which is uppercase equivalent of the supplied varying string.*
- TYPE(VARYING\_STRING) VSTRING\_TO\_UPPERCASE\_C (STRING)  
*Returns a varying string which is uppercase equivalent of the supplied character string.*
- TYPE(VARYING\_STRING) VSTRING\_TO\_UPPERCASE\_VS (STRING)  
*Returns a varying string which is uppercase equivalent of the supplied varying string.*

### 6.43.1 Detailed Description

This module contains all string manipulation and transformation routines.

### 6.43.2 Function Documentation

#### 6.43.2.1 function STRINGS::CHARACTER\_TO\_LOWERCASE\_C (CHARACTER(LEN=\*)**intent(in)** STRING) [private]

**Parameters:**

**STRING** The string to convert to lowercase

Definition at line 1609 of file strings.f90.

References IS\_UPPERCASE().

Here is the call graph for this function:

#### 6.43.2.2 function STRINGS::CHARACTER\_TO\_LOWERCASE\_VS (TYPE(VARYING\_STRING)**intent(in)** STRING) [private]

Returns a character string that is the lowercase equivalent of the supplied varying string.

**Parameters:**

**STRING** The string to convert

Definition at line 1634 of file strings.f90.

References IS\_UPPERCASE().

Here is the call graph for this function:

**6.43.2.3 function STRINGS::CHARACTER\_TO\_UPPERCASE\_C  
(CHARACTER(LEN=\*),intent(in) STRING) [private]**

Returns a character string which is uppercase equivalent of the supplied character string.

**Parameters:**

**STRING** The string to convert

Definition at line 1709 of file strings.f90.

References IS\_LOWER CASE().

Here is the call graph for this function:

**6.43.2.4 function STRINGS::CHARACTER\_TO\_UPPERCASE\_VS  
(TYPE(VARYING\_STRING),intent(in) STRING) [private]**

Returns a character string which is uppercase equivalent of the supplied varying string.

**Parameters:**

**STRING** The string to convert

Definition at line 1734 of file strings.f90.

References IS\_LOWER CASE().

Here is the call graph for this function:

**6.43.2.5 LOGICAL STRINGS::IS\_ABBREVIATION\_C\_C (CHARACTER(LEN=\*),intent(in)  
SHORT, CHARACTER(LEN=\*),intent(in) LONG, INTEGER(INTG),intent(in)  
MIN\_NUM\_CHARACTERS) [private]**

**IS\_ABBREVIATION** returns .TRUE. if the character string SHORT is an abbreviation of the character string LONG. SHORT must be at least MIN\_NUM\_CHARACTERS long.

**Parameters:**

**SHORT** The short form of the string

**LONG** The long form of the string

**MIN\_NUM\_CHARACTERS** The minimum number of characters to match

Definition at line 160 of file strings.f90.

**6.43.2.6 LOGICAL STRINGS::IS\_ABBREVIATION\_C\_VS (CHARACTER(LEN=\*),intent(in)  
SHORT, TYPE(VARYING\_STRING),intent(in) LONG, INTEGER(INTG),intent(in)  
MIN\_NUM\_CHARACTERS) [private]**

**IS\_ABBREVIATION** returns .TRUE. if the character string SHORT is an abbreviation of the varying string LONG. SHORT must be at least MIN\_NUM\_CHARACTERS long.

**Parameters:**

**SHORT** The short form of the string

**LONG** The long form of the string

**MIN\_NUM\_CHARACTERS** The minimum number of characters to match

Definition at line 192 of file strings.f90.

**6.43.2.7 LOGICAL STRINGS::IS\_ABBREVIATION\_VS\_C (TYPE(VARYING\_STRING),intent(in) *SHORT*, CHARACTER(LEN=\*),intent(in) *LONG*, INTEGER(INTG),intent(in) *MIN\_NUM\_CHARACTERS*) [private]**

**IS\_ABBREVIATION** returns .TRUE. if the varying string *SHORT* is an abbreviation of the character string *LONG*. *SHORT* must be at least *MIN\_NUM\_CHARACTERS* long.

**Parameters:**

**SHORT** The short form of the string

**LONG** The long form of the string

**MIN\_NUM\_CHARACTERS** The minimum number of characters to match

Definition at line 224 of file strings.f90.

**6.43.2.8 LOGICAL STRINGS::IS\_ABBREVIATION\_VS\_VS (TYPE(VARYING\_STRING),intent(in) *SHORT*, TYPE(VARYING\_STRING),intent(in) *LONG*, INTEGER(INTG),intent(in) *MIN\_NUM\_CHARACTERS*) [private]**

**IS\_ABBREVIATION** returns .TRUE. if the varying string *SHORT* is an abbreviation of the varying string *LONG*. *SHORT* must be at least *MIN\_NUM\_CHARACTERS* long.

**Parameters:**

**SHORT** The short form of the string

**LONG** The long form of the string

**MIN\_NUM\_CHARACTERS** The minimum number of characters to match

Definition at line 256 of file strings.f90.

**6.43.2.9 LOGICAL STRINGS::IS\_DIGIT (CHARACTER(LEN=1),intent(in) *CHARAC*)**

**IS\_DIGIT** returns .TRUE. if the character *CHARAC* is a digit character (i.e. 0..9).

**Parameters:**

**CHARAC** The character to test if it is a digit

Definition at line 287 of file strings.f90.

**6.43.2.10 LOGICAL STRINGS::IS\_LETTER (CHARACTER(LEN=1),intent(in) *CHARAC*)**

**IS\_LETTER** returns .TRUE. if the character *CHARAC* is a letter character (i.e. A..Z or a..z).

**Parameters:**

**CHARAC** The character to test if it is a letter

Definition at line 305 of file strings.f90.

#### 6.43.2.11 LOGICAL STRINGS::IS\_LOWERCase (CHARACTER(LEN=1),intent(in) *CHARC*) [private]

Returns .TRUE. if the supplied character is a lowercase character.

**Parameters:**

*CHARC* The character to test if it is lowercase

Definition at line 324 of file strings.f90.

Referenced by CHARACTER\_TO\_UPPERCASE\_C(), CHARACTER\_TO\_UPPERCASE\_VS(), VSTRING\_TO\_UPPERCASE\_C(), and VSTRING\_TO\_UPPERCASE\_VS().

Here is the caller graph for this function:

#### 6.43.2.12 LOGICAL STRINGS::IS\_UPPERCase (CHARACTER(LEN=1),intent(in) *CHARC*) [private]

Returns .TRUE. if the supplied character is an uppercase character.

**Parameters:**

*CHARC* The character to test if it is uppercase

Definition at line 346 of file strings.f90.

Referenced by CHARACTER\_TO\_LOWERCase\_C(), CHARACTER\_TO\_LOWERCase\_VS(), VSTRING\_TO\_LOWERCase\_C(), and VSTRING\_TO\_LOWERCase\_VS().

Here is the caller graph for this function:

#### 6.43.2.13 LOGICAL STRINGS::IS\_WHITESPACE (CHARACTER(LEN=1),intent(in) *CHARAC*)

IS\_WHITESPACE returns .TRUE. if the character CHARAC is a whitespace character (i.e. space, tabs, etc.).

**Parameters:**

*CHARAC* The character to test if it is whitespace

Definition at line 368 of file strings.f90.

#### 6.43.2.14 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST\_TO\_- CHARACTER\_C (INTEGER(INTG),intent(in) *NUMBER\_IN\_LIST*, CHARACTER(LEN=\*) ,dimension(*number\_in\_list*),intent(in) *LIST*, CHARACTER(LEN=\*) ,intent(in) *FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, INTEGER(INTG),dimension(*number\_in\_list*),intent(in),optional *LIST\_LENGTHS*) [private]

Converts a character list to its equivalent character string representation as determined by the supplied format. If present, the optional argument LIST\_LENGTHS is used for the lengths of each list elements length otherwise the trimmed length is used. NOTE: The FORMAT is ignored for this child FUNCTION.

**Parameters:**

**NUMBER\_IN\_LIST** The number of items in the list  
**LIST** LIST(i). The i'th item in the list  
**FORMAT** The format to use. Ignored for character lists.  
**ERR** The error code  
**ERROR** The error string  
**LIST\_LENGTHS** LIST\_LENGTHS(i). Optional, The length of the i'th list item.

Definition at line 387 of file strings.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.43.2.15 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST\_TO\_CHARACTER\_DP**  
 (INTEGER(INTG),intent(in) **NUMBER\_IN\_LIST**, REAL(DP),dimension(number\_-in\_list),intent(in) **LIST**, CHARACTER(LEN=\*),intent(in) **FORMAT**,  
 INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**)  
 [private]

Converts a double precision list to its equivalent character string representation as determined by the supplied format string.

**Parameters:**

**NUMBER\_IN\_LIST** The number of items in the list  
**LIST** LIST(i). The i'th item in the list  
**FORMAT** The format to use for the conversion  
**ERR** The error code  
**ERROR** The error string

Definition at line 668 of file strings.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    BASE\_-ROUTINES::EXITS(), and NUMBER\_TO\_CHARACTER\_DP().

Here is the call graph for this function:

**6.43.2.16 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST\_TO\_-CHARACTER\_INTG** (INTEGER(INTG),intent(in) **NUMBER\_IN\_LIST**,  
 INTEGER(INTG),dimension(number\_in\_list),intent(in) **LIST**,  
 CHARACTER(LEN=\*),intent(in) **FORMAT**, INTEGER(INTG),intent(out) **ERR**,  
 TYPE(VARYING\_STRING),intent(out) **ERROR**)   [private]

Converts an integer list to its equivalent character string representation as determined by the supplied format.

**Parameters:**

**NUMBER\_IN\_LIST** The number of items in the list  
**LIST** LIST(i). The i'th item in the list

**FORMAT** The format to use for the conversion

**ERR** The error code

**ERROR** The error string

Definition at line 458 of file strings.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `NUMBER_TO_CHARACTER_INTG()`.

Here is the call graph for this function:

**6.43.2.17 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST\_TO\_CHARACTER\_L  
(INTEGER(INTG),intent(in) NUMBER\_IN\_LIST, LOGICAL,dimension(number\_in\_list),intent(in) LIST, CHARACTER(LEN=\*),intent(in) FORMAT,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR)  
[private]**

Converts a logical list to its equivalent character string representation as determined by the supplied format string. The FORMAT is ignored for this child FUNCTION.

#### Parameters:

**NUMBER\_IN\_LIST** The number of items in the list

**LIST** LIST(i). The i'th item in the list

**FORMAT** The format to use. Ignored for logical lists.

**ERR** The error code

**ERROR** The error string

Definition at line 564 of file strings.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `LOGICAL_TO_CHARACTER()`.

Here is the call graph for this function:

**6.43.2.18 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST\_TO\_CHARACTER\_LINTG  
(INTEGER(INTG),intent(in) NUMBER\_IN\_LIST,  
INTEGER(LINTG),dimension(number\_in\_list),intent(in) LIST,  
CHARACTER(LEN=\*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

Converts an long integer list to its equivalent character string representation as determined by the supplied format.

#### Parameters:

**NUMBER\_IN\_LIST** The number of items in the list

**LIST** LIST(i). The i'th item in the list

**FORMAT** The format to use for the conversion

**ERR** The error code

**ERROR** The error string

Definition at line 511 of file strings.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `NUMBER_TO_CHARACTER_LINTG()`.

Here is the call graph for this function:

**6.43.2.19 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST\_TO\_CHARACTER\_SP**  
`(INTEGER(INTG),intent(in) NUMBER_IN_LIST, REAL(SP),dimension(number_in_list),intent(in) LIST, CHARACTER(LEN=*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)`  
`[private]`

Converts a single precision list to its equivalent character string representation as determined by the supplied format string.

#### Parameters:

**NUMBER\_IN\_LIST** The number of items in the list  
**LIST** `LIST(i)`. The i'th item in the list  
**FORMAT** The format to use for the conversion  
**ERR** The error code  
**ERROR** The error string

Definition at line 616 of file strings.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `NUMBER_TO_CHARACTER_SP()`.

Here is the call graph for this function:

**6.43.2.20 CHARACTER(LEN=MAXSTRLEN) STRINGS::LOGICAL\_TO\_CHARACTER**  
`(LOGICAL,intent(in) LOGICALVALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)`

Converts a logical value to either a "TRUE" or "FALSE" character string.

#### Parameters:

**LOGICALVALUE** The logical value to convert  
**ERR** The error code  
**ERROR** The error string

Definition at line 720 of file strings.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `LIST_TO_CHARACTER_L()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.43.2.21 TYPE(VARYING\_STRING) STRINGS::LOGICAL\_TO\_VSTRING  
 (LOGICAL,intent(in) LOGICALVALUE, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

Converts a logical value to either a "TRUE" or "FALSE" varying string.

**Parameters:**

**LOGICALVALUE** The logical value to convert  
**ERR** The error code  
**ERROR** The error string

Definition at line 750 of file strings.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by INPUT\_OUTPUT::WR(), INPUT\_OUTPUT::WRITE\_STRING\_FMT\_VALUE\_L(), INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_C\_L(), INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_DP\_L(), INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_INTG\_L(), INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_L\_C(), INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_L\_DP(), INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_L\_INTG(), INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_L\_L(), INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_L\_SP(), INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_L\_VS(), INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_SP\_L(), INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_VS\_L(), and INPUT\_OUTPUT::WRITE\_STRING\_VALUE\_L().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.43.2.22 CHARACTER(LEN=MAXSTRLEN) STRINGS::NUMBER\_TO\_CHARACTER\_DP  
 (REAL(DP),intent(in) NUMBER, CHARACTER(LEN=\*),intent(in) FORMAT,  
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR)  
 [private]**

Converts a double precision number to its equivalent character string representation as determined by the supplied format string. Note If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="\*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.

**Parameters:**

**NUMBER** The number to convert  
**FORMAT** The format to use in the conversion  
**ERR** The error code  
**ERROR** The error string

Definition at line 947 of file strings.f90.

References KINDS::\_DP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by LIST\_TO\_CHARACTER\_DP().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.43.2.23 CHARACTER(LEN=MAXSTRLEN) STRINGS::NUMBER\_TO\_CHARACTER\_INTG (INTEGER(INTG),intent(in) *NUMBER*, CHARACTER(LEN=\*),intent(in) *FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]**

Converts an integer number to its equivalent character string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g. "I3".

**Parameters:**

***NUMBER*** The number to convert

***FORMAT*** The format to use in the conversion

***ERR*** The error code

***ERROR*** The error string

Definition at line 780 of file strings.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Referenced by LIST\_TO\_CHARACTER\_INTG().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.43.2.24 CHARACTER(LEN=MAXSTRLEN) STRINGS::NUMBER\_TO\_CHARACTER\_LINTG (INTEGER(LINTG),intent(in) *NUMBER*, CHARACTER(LEN=\*),intent(in) *FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]**

Converts a long integer number to its equivalent character string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g. "I3".

**Parameters:**

***NUMBER*** The number to convert

***FORMAT*** The format to use in the conversion

***ERR*** The error code

***ERROR*** The error string

Definition at line 817 of file strings.f90.

References    BASE\_ROUTINES::ENTERS(),    BASE\_ROUTINES::ERRORS(),    and    BASE\_-ROUTINES::EXITS().

Referenced by LIST\_TO\_CHARACTER\_LINTG().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.43.2.25 CHARACTER(LEN=MAXSTRLEN) STRINGS::NUMBER\_TO\_CHARACTER\_SP  
(REAL(SP),intent(in) NUMBER, CHARACTER(LEN=\*),intent(in) FORMAT,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR)  
[private]**

Converts a single precision number to its equivalent character string representation as determined by the supplied format string. NOTE: If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="\*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.

**Parameters:**

**NUMBER** The number to convert

**FORMAT** The format to use in the conversion

**ERR** The error code

**ERROR** The error string

Definition at line 854 of file strings.f90.

References KINDS::\_SP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by LIST\_TO\_CHARACTER\_SP().

Here is the call graph for this function:

Here is the caller graph for this function:

---

**6.43.2.26 TYPE(VARYING\_STRING) STRINGS::NUMBER\_TO\_VSTRING\_DP  
(REAL(DP),intent(in) NUMBER, CHARACTER(LEN=\*),intent(in) FORMAT,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR)  
[private]**

Converts a double precision number to its equivalent varying string representation as determined by the supplied format string. Note If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="\*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.

**Parameters:**

**NUMBER** The number to convert

**FORMAT** The format to use in the conversion

**ERR** The error code

**ERROR** The error string

Definition at line 1222 of file strings.f90.

References KINDS::\_DP, and BASE\_ROUTINES::ERRORS().

Here is the call graph for this function:

---

**6.43.2.27** **TYPE(VARYING\_STRING) STRINGS::NUMBER\_TO\_VSTRING\_INTG**  
**(INTEGER(INTG),intent(in) NUMBER, CHARACTER(LEN=\*),intent(in) FORMAT,**  
**INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR)**  
**[private]**

Converts an integer number to its equivalent varying string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g. "I3".

**Parameters:**

**NUMBER** The number to convert  
**FORMAT** The format to use in the conversion  
**ERR** The error code  
**ERROR** The error string

Definition at line 1039 of file strings.f90.

References BASE\_ROUTINES::ERRORS().

Here is the call graph for this function:

**6.43.2.28** **TYPE(VARYING\_STRING) STRINGS::NUMBER\_TO\_VSTRING\_LINTG**  
**(INTEGER(LINTG),intent(in) NUMBER, CHARACTER(LEN=\*),intent(in) FORMAT,**  
**INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR)**  
**[private]**

Converts a long integer number to its equivalent varying string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g., "I3".

**Parameters:**

**NUMBER** The number to convert  
**FORMAT** The format to use in the conversion  
**ERR** The error code  
**ERROR** The error string

Definition at line 1082 of file strings.f90.

References BASE\_ROUTINES::ERRORS().

Here is the call graph for this function:

**6.43.2.29** **TYPE(VARYING\_STRING) STRINGS::NUMBER\_TO\_VSTRING\_SP**  
**(REAL(SP),intent(in) NUMBER, CHARACTER(LEN=\*),intent(in) FORMAT,**  
**INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR)**  
**[private]**

Converts a single precision number to its equivalent varying string representation as determined by the supplied format string. Note If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="\*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.

**Parameters:**

**NUMBER** The number to convert

**FORMAT** The format to use in the conversion

**ERR** The error code

**ERROR** The error string

Definition at line 1125 of file strings.f90.

References KINDS::\_SP, BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.43.2.30 REAL(DP) STRINGS::STRING\_TO\_DOUBLE\_C (CHARACTER(LEN=\*),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

Converts a character string representation of a number to a double precision number.

**Parameters:**

**STRING** The string to convert

**ERR** The error code !<The error code

**ERROR** The error string !<The error string

Definition at line 1322 of file strings.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.43.2.31 REAL(DP) STRINGS::STRING\_TO\_DOUBLE\_VS (TYPE(VARYING\_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

Converts a varying string representation of a number to a double precision number.

**Parameters:**

**STRING** The string to convert

**ERR** The error code

**ERROR** The error string

Definition at line 1349 of file strings.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.43.2.32 INTEGER(INTG) STRINGS::STRING\_TO\_INTEGER\_C  
 (CHARACTER(LEN=\*),intent(in) STRING, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

Converts a character string representation of a number to an integer.

**Parameters:**

**STRING** The string to convert

**ERR** The error code

**ERROR** The error string

Definition at line 1380 of file strings.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.43.2.33 INTEGER(INTG) STRINGS::STRING\_TO\_INTEGER\_VS (TYPE(VARYING\_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

Converts a varying string representation of a number to an integer.

**Parameters:**

**STRING** The string to convert

**ERR** The error code

**ERROR** The error string

Definition at line 1407 of file strings.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.43.2.34 LOGICAL STRINGS::STRING\_TO\_LOGICAL\_C (CHARACTER(LEN=\*),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

Converts a character string representation of a boolean (TRUE or FALSE) to a logical.

**Parameters:**

**STRING** The string to convert

**ERR** The error code

**ERROR** The error string

Definition at line 1496 of file strings.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.43.2.35 LOGICAL STRINGS::STRING\_TO\_LOGICAL\_VS (TYPE(VARYING\_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

Converts a varying string representation of a boolean (TRUE or FALSE) to a logical.

**Parameters:**

**STRING** The string to convert

**ERR** The error code

**ERROR** The error string

Definition at line 1523 of file strings.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.43.2.36 INTEGER(LINTG) STRINGS::STRING\_TO\_LONG\_INTEGER\_C (CHARACTER(LEN=\*),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

Converts a character string representation of a number to a long integer.

**Parameters:**

**STRING** The string to convert

**ERR** The error code

**ERROR** The error string

Definition at line 1438 of file strings.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.43.2.37 INTEGER(LINTG) STRINGS::STRING\_TO\_LONG\_INTEGER\_VS (TYPE(VARYING\_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

Converts a varying string representation of a number to a long integer.

**Parameters:**

**STRING** The string to convert

**ERR** The error code

**ERROR** The error string

Definition at line 1465 of file strings.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

---

**6.43.2.38 REAL(SP) STRINGS::STRING\_TO\_SINGLE\_C (CHARACTER(LEN=\*)*,intent(in)* STRING, INTEGER(INTG)*,intent(out)* ERR, TYPE(VARYING\_STRING)*,intent(out)* ERROR) [private]**

Converts a character string representation of a number to a single precision number.

**Parameters:**

**STRING** The string to convert

**ERR** The error code

**ERROR** The error string

Definition at line 1552 of file strings.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.43.2.39 REAL(SP) STRINGS::STRING\_TO\_SINGLE\_VS (TYPE(VARYING\_STRING)*,intent(in)* STRING, INTEGER(INTG)*,intent(out)* ERR, TYPE(VARYING\_STRING)*,intent(out)* ERROR) [private]**

Converts a varying string representation of a number to a single precision number.

**Parameters:**

**STRING** The string to convert

**ERR** The error code

**ERROR** The error string

Definition at line 1579 of file strings.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Here is the call graph for this function:

**6.43.2.40 TYPE(VARYING\_STRING) STRINGS::VSTRING\_TO\_LOWERCASE\_C (CHARACTER(LEN=\*)*,intent(in)* STRING) [private]**

Returns a varying string that is the lowercase equivalent of the supplied character string.

**Parameters:**

**STRING** The string to convert

Definition at line 1659 of file strings.f90.

References IS\_UPPERCASE().

Here is the call graph for this function:

**6.43.2.41 TYPE(VARYING\_STRING) STRINGS::VSTRING\_TO\_LOWERCASE\_VS  
(TYPE(VARYING\_STRING),intent(in) STRING) [private]**

Returns a varying string that is the lowercase equivalent of the supplied varying string.

**Parameters:**

*STRING* The string to convert

Definition at line 1684 of file strings.f90.

References IS\_UPPERCASE().

Here is the call graph for this function:

**6.43.2.42 TYPE(VARYING\_STRING) STRINGS::VSTRING\_TO\_UPPERCASE\_C  
(CHARACTER(LEN=\*),intent(in) STRING) [private]**

Returns a varying string which is uppercase equivalent of the supplied character string.

**Parameters:**

*STRING* The string to convert

Definition at line 1759 of file strings.f90.

References IS\_LOWERCASE().

Here is the call graph for this function:

**6.43.2.43 TYPE(VARYING\_STRING) STRINGS::VSTRING\_TO\_UPPERCASE\_VS  
(TYPE(VARYING\_STRING),intent(in) STRING) [private]**

Returns a varying string which is uppercase equivalent of the supplied varying string.

**Parameters:**

*STRING* The string to convert

Definition at line 1784 of file strings.f90.

References IS\_LOWERCASE().

Here is the call graph for this function:

## 6.44 TIMER Namespace Reference

This module contains routines for timing the program.

### Classes

- interface [interface](#)

### Functions

- subroutine [CPU\\_TIMER](#) (TIME\_TYPE, TIME, ERR, ERROR,\*)

### Variables

- INTEGER(INTG), parameter [USER\\_CPU](#) = 1
- INTEGER(INTG), parameter [SYSTEM\\_CPU](#) = 2
- INTEGER(INTG), parameter [TOTAL\\_CPU](#) = 3

#### 6.44.1 Detailed Description

This module contains routines for timing the program.

#### 6.44.2 Function Documentation

##### 6.44.2.1 subroutine [TIMER::CPU\\_TIMER](#) (INTEGER(INTG),intent(in) TIME\_TYPE, REAL(SP),dimension(\*),intent(out) TIME, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)

Definition at line 90 of file timer\_f.f90.

References F90C::C2FSTRING(), BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_ROUTINES::EXITS().

Referenced by [TIMER::interface::CPUTIMER\(\)](#), [SOLVER\\_ROUTINES::SOLVER\\_MATRICES\\_ASSEMBLE\(\)](#), and [SOLVER\\_ROUTINES::SOLVER\\_SOLVE\(\)](#).

Here is the call graph for this function:

Here is the caller graph for this function:

#### 6.44.3 Variable Documentation

##### 6.44.3.1 INTEGER(INTG),parameter [TIMER::SYSTEM\\_CPU](#) = 2

Definition at line 62 of file timer\_f.f90.

##### 6.44.3.2 INTEGER(INTG),parameter [TIMER::TOTAL\\_CPU](#) = 3

Definition at line 63 of file timer\_f.f90.

**6.44.3.3 INTEGER(INTG),parameter TIMER::USER\_CPU = 1**

Definition at line 61 of file timer\_f.f90.

## 6.45 TREES Namespace Reference

Implements trees of base types.

### Classes

- struct [TREE\\_NODE\\_TYPE](#)
- struct [TREE\\_TYPE](#)

### Functions

- subroutine [TREE\\_CREATE\\_FINISH](#) (TREE, ERR, ERROR,\*)
 

*Finishes the creation of a tree created with TREE\_CREATE\_START.*
- subroutine [TREE\\_CREATE\\_START](#) (TREE, ERR, ERROR,\*)
 

*Starts the creation of a tree and returns a pointer to the created tree.*
- subroutine [TREE\\_DESTROY](#) (TREE, ERR, ERROR,\*)
 

*Destroys a tree.*
- subroutine [TREE\\_DETACH\\_AND\\_DESTROY](#) (TREE, NUMBER\_IN\_TREE, TREE\_VALUES, ERR, ERROR,\*)
 

*Detaches the tree values and returns them as a pointer to the an array and then destroys the tree.*
- subroutine [TREE\\_DETACH\\_IN\\_ORDER](#) (TREE, X, COUNT, TREE\_VALUES, ERR, ERROR,\*)
 

*Detaches the tree values in order from the specified tree node and adds them to the tree values array.*
- subroutine [TREE\\_FINALISE](#) (TREE, ERR, ERROR,\*)
 

*Finalises a tree and deallocates all memory.*
- subroutine [TREE\\_INITIALISE](#) (TREE, ERR, ERROR,\*)
 

*Initialises a tree.*
- subroutine [TREE\\_INSERT\\_TYPE\\_SET](#) (TREE, INSERT\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the insert type for a tree.*
- subroutine [TREE\\_ITEM\\_DELETE](#) (TREE, KEY, ERR, ERROR,\*)
 

*Deletes a tree node specified by a key from a tree.*
- subroutine [TREE\\_ITEM\\_INSERT](#) (TREE, KEY, VALUE, INSERT\_STATUS, ERR, ERROR,\*)
 

*Inserts a tree node into a red-black tree.*
- subroutine [TREE\\_NODE\\_FINALISE](#) (TREE, TREE\_NODE, ERR, ERROR,\*)
 

*Finalises a tree node and deallocates all memory.*
- subroutine [TREE\\_NODE\\_INITIALISE](#) (TREE, TREE\_NODE, ERR, ERROR,\*)
 

*Initialises a tree node.*

- subroutine `TREE_NODE_KEY_GET` (TREE, TREE\_NODE, KEY, ERR, ERROR,\*)
 

*Gets the key at a specified tree node.*
- subroutine `TREE_NODE_VALUE_GET` (TREE, TREE\_NODE, VALUE, ERR, ERROR,\*)
 

*Gets the value at a specified tree node.*
- subroutine `TREE_NODE_VALUE_SET` (TREE, TREE\_NODE, VALUE, ERR, ERROR,\*)
 

*Sets the value at a specified tree node.*
- subroutine `TREE_OUTPUT` (ID, TREE, ERR, ERROR,\*)
 

*Outputs a tree to the specified output stream ID.*
- subroutine `TREE_OUTPUT_IN_ORDER` (ID, TREE, X, ERR, ERROR,\*)
 

*Outputs a tree in order to the specified output stream ID from the specified tree node.*
- `TYPE(TREE_NODE_TYPE)`, pointer `TREE_PREDECESSOR` (TREE, X, ERR, ERROR)
 

*Returns the predecessor of a tree at a specified tree node.*
- subroutine `TREE_SEARCH` (TREE, KEY, X, ERR, ERROR,\*)
 

*Searches a tree to see if it contains a key.*
- `TYPE(TREE_NODE_TYPE)`, pointer `TREE_SUCCESSOR` (TREE, X, ERR, ERROR)
 

*Returns the successor of a tree at a specified tree node.*

## Variables

- `INTEGER(INTG)`, parameter `TREE_BLACK_NODE` = 0
 

*The black colour type for a tree node.*
- `INTEGER(INTG)`, parameter `TREE_RED_NODE` = 1
 

*The red colour type for a tree node.*
- `INTEGER(INTG)`, parameter `TREE_NODE_INSERT_SUCESSFUL` = 1
 

*Successful insert status.*
- `INTEGER(INTG)`, parameter `TREE_NODE_DUPLICATE_KEY` = 2
 

*Duplicate key found for those trees that do not allow duplicate keys.*
- `INTEGER(INTG)`, parameter `TREE_DUPLICATES_ALLOWED_TYPE` = 1
 

*Duplicate keys allowed tree type.*
- `INTEGER(INTG)`, parameter `TREE_NO_DUPLICATES_ALLOWED` = 2
 

*No duplicate keys allowed tree type.*

### 6.45.1 Detailed Description

Implements trees of base types.

## 6.45.2 Function Documentation

### 6.45.2.1 subroutine TREES::TREE\_CREATE\_FINISH (TYPE(TREE\_TYPE),pointer *TREE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Finishes the creation of a tree created with TREE\_CREATE\_START.

**See also:**

[\(TREES::TREE\\_CREATE\\_START\).](#)

**Parameters:**

***TREE*** A pointer to the tree to finish

***ERR*** The error code

***ERROR*** The error string.

Definition at line 123 of file trees.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_- ROUTINES::EXITS(), TREE\_FINALISE(), and TREE\_NODE\_INITIALISE().

Referenced by MESH\_ROUTINES::MESH\_TOPOLOGY\_NODES\_CALCULATE(), and NODE\_- ROUTINES::NODES\_CREATE\_START().

Here is the call graph for this function:

Here is the caller graph for this function:

### 6.45.2.2 subroutine TREES::TREE\_CREATE\_START (TYPE(TREE\_TYPE),pointer *TREE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Starts the creation of a tree and returns a pointer to the created tree.

**See also:**

[\(TREES::TREE\\_CREATE\\_FINISH\).](#)

**Parameters:**

***TREE*** A pointer to the tree to create. Must not be associated on entry.

***ERR*** The error code.

***ERROR*** The error string.

Definition at line 167 of file trees.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_- ROUTINES::EXITS(), TREE\_DUPLICATES\_ALLOWED\_TYPE, TREE\_FINALISE(), and TREE\_- INITIALISE().

Referenced by MESH\_ROUTINES::MESH\_TOPOLOGY\_NODES\_CALCULATE(), and NODE\_- ROUTINES::NODES\_CREATE\_START().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.45.2.3 subroutine TREES::TREE\_DESTROY (TYPE(TREE\_TYPE),pointer *TREE*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Destroys a tree.

**Parameters:**

***TREE*** A pointer to the tree to destroy

***ERR*** The error code

***ERROR*** The error string.

Definition at line 200 of file trees.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_-ROUTINES::EXITS(), and TREE\_FINALISE().

Referenced by MESH\_ROUTINES::MESH\_TOPOLOGY\_NODES\_CALCULATE(), and NODE\_-ROUTINES::NODES\_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.45.2.4 subroutine TREES::TREE\_DETACH\_AND\_DESTROY (TYPE(TREE\_-TYPE),pointer *TREE*, INTEGER(INTG),intent(out) *NUMBER\_IN\_TREE*,  
INTEGER(INTG),dimension(:),pointer *TREE\_VALUES*, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Detaches the tree values and returns them as a pointer to the an array and then destroys the tree.

**Parameters:**

***TREE*** A pointer to the tree to detach and destroy

***NUMBER\_IN\_TREE*** On exit, the number in the array that has been detached

***TREE\_VALUES*** On exit, a pointer to the dettached tree values. Must not be associated on entry.

***ERR*** The error code

***ERROR*** The error string.

Definition at line 228 of file trees.f90.

References      BASE\_ROUTINES::ENTERS(),      BASE\_ROUTINES::ERRORS(),      BASE\_-ROUTINES::EXITS(), TREE\_DETACH\_IN\_ORDER(), and TREE\_FINALISE().

Referenced by MESH\_ROUTINES::MESH\_TOPOLOGY\_NODES\_CALCULATE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.45.2.5 subroutine TREES::TREE\_DETACH\_IN\_ORDER (TYPE(TREE\_TYPE),pointer  
*TREE*, TYPE(TREE\_NODE\_TYPE),pointer *X*, INTEGER(INTG),intent(inout)  
*COUNT*, INTEGER(INTG),dimension(:),intent(inout) *TREE\_VALUES*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)  
[private]**

Detaches the tree values in order from the specified tree node and adds them to the tree values array.

**Parameters:**

**TREE** A pointer to the tree to detach  
**X** A pointer to the specified tree node to detach from  
**COUNT** The current number in the detached tree values array  
**TREE\_VALUES** The current detached tree values array  
**ERR** The error code  
**ERROR** The error string.

Definition at line 273 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `TREE_DETACH_AND_DESTROY()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.45.2.6 subroutine TREES::TREE\_FINALISE (TYPE(TREE\_TYPE),pointer TREE,  
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)  
 [private]**

Finalises a tree and deallocates all memory.

**Parameters:**

**TREE** A pointer to the tree to finalise  
**ERR** The error code  
**ERROR** The error string.

Definition at line 317 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `TREE_NODE_FINALISE()`.

Referenced by `TREE_CREATE_FINISH()`, `TREE_CREATE_START()`, `TREE_DESTROY()`, and `TREE_DETACH_AND_DESTROY()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.45.2.7 subroutine TREES::TREE\_INITIALISE (TYPE(TREE\_TYPE),pointer TREE,  
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)  
 [private]**

Initialises a tree.

**Parameters:**

**TREE** A pointer to the tree to initialise  
**ERR** The error code  
**ERROR** The error string.

Definition at line 345 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `TREE_CREATE_START()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.45.2.8 subroutine TREES::TREE\_INSERT\_TYPE\_SET (TYPE(TREE\_TYPE),pointer *TREE*, INTEGER(INTG),intent(in) *INSERT\_TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Sets/changes the insert type for a tree.

**Parameters:**

*TREE* A pointer to the tree

*INSERT\_TYPE* The insert type to set

**See also:**

[TREES::TreeInsertTypes](#), [TREES::TreeInsertTypes](#)

*ERR* The error code

*ERROR* The error string.

Definition at line 377 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `TREE_DUPLICATES_ALLOWED_TYPE`, and `TREE_NO_DUPLICATES_ALLOWED`.

Referenced by `MESH_ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE()`, and `NODE_ROUTINES::NODES_CREATE_START()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.45.2.9 subroutine TREES::TREE\_ITEM\_DELETE (TYPE(TREE\_TYPE),pointer *TREE*, INTEGER(INTG),intent(in) *KEY*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Deletes a tree node specified by a key from a tree.

**Parameters:**

*TREE* A pointer to the Red-Black tree to delete from

*KEY* A pointer to the tree node to delete

*ERR* The error code

*ERROR* The error string.

Definition at line 419 of file trees.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `TREE_BLACK_NODE`, `TREE_RED_NODE`, and `TREE_SUCCESSOR()`.

Referenced by `NODE_ROUTINES::NODE_NUMBER_SET()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.45.2.10 subroutine `TREES::TREE_ITEM_INSERT` (`TYPE(TREE_TYPE),pointer TREE, INTEGER(INTG),intent(in) KEY, INTEGER(INTG),intent(in) VALUE, INTEGER(INTG),intent(out) INSERT_STATUS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, */)`**

Inserts a tree node into a red-black tree.

**Parameters:**

**`TREE`** A pointer to the Red-Black tree to insert into

**`KEY`** The key to insert

**`VALUE`** The value to insert

**`INSERT_STATUS`** On exit, the status of the insert

**See also:**

[TREES::TreeNodeInsertStatus](#),[TREES::TreeNodeInsertStatus](#)

**`ERR`** The error code

**`ERROR`** The error string.

Definition at line 668 of file trees.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, `TREE_BLACK_NODE`, `TREE_NO_DUPLICATES_ALLOWED`, `TREE_NODE_DUPLICATE_KEY`, `TREE_NODE_INITIALISE()`, `TREE_NODE_INSERT_SUCESSFUL`, and `TREE_RED_NODE`.

Referenced by `MESH_ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE()`, `NODE_ROUTINES::NODE_NUMBER_SET()`, and `NODE_ROUTINES::NODES_CREATE_START()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.45.2.11 subroutine `TREES::TREE_NODE_FINALISE` (`TYPE(TREE_TYPE),pointer TREE, TYPE(TREE_NODE_TYPE),pointer TREE_NODE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, */ [private]`)**

Finalises a tree node and deallocates all memory.

**Parameters:**

**`TREE`** A pointer to the tree containing the tree node to finalise

**`TREE_NODE`** A pointer to the tree node to finalise

**`ERR`** The error code

**`ERROR`** The error string.

Definition at line 853 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `TREE_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.45.2.12 subroutine TREES::TREE\_NODE\_INITIALISE (TYPE(TREE\_TYPE),pointer *TREE*,  
TYPE(TREE\_NODE\_TYPE),pointer *TREE\_NODE*, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Initialises a tree node.

**Parameters:**

`TREE` A pointer to the tree containing the tree node to initialise

`TREE_NODE` A pointer to the tree node to initialise

`ERR` The error code

`ERROR` The error string.

Definition at line 886 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `TREE_BLACK_NODE`.

Referenced by `TREE_CREATE_FINISH()`, and `TREE_ITEM_INSERT()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.45.2.13 subroutine TREES::TREE\_NODE\_KEY\_GET (TYPE(TREE\_TYPE),pointer *TREE*,  
TYPE(TREE\_NODE\_TYPE),pointer *TREE\_NODE*, INTEGER(INTG),intent(out)  
*KEY*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out)  
*ERROR*, \*)**

Gets the key at a specified tree node.

**Parameters:**

`TREE` A pointer to the tree containing the tree node

`TREE_NODE` A pointer to the tree node to get the key of

`KEY` On exit, the key of the specified tree node

`ERR` The error code

`ERROR` The error string.

Definition at line 924 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

---

**6.45.2.14 subroutine TREES::TREE\_NODE\_VALUE\_GET (TYPE(TREE\_TYPE),pointer *TREE*,  
TYPE(TREE\_NODE\_TYPE),pointer *TREE\_NODE*, INTEGER(INTG),intent(out)  
*VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out)  
*ERROR*, \*)**

Gets the value at a specified tree node.

**Parameters:**

*TREE* A pointer to the tree containing the tree node  
*TREE\_NODE* A pointer to the tree node to get the value of  
*VALUE* On exit, the value of the specified tree node  
*ERR* The error code  
*ERROR* The error string.

Definition at line 962 of file trees.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Referenced by NODE\_ROUTINES::NODE\_CHECK\_EXISTS().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.45.2.15 subroutine TREES::TREE\_NODE\_VALUE\_SET (TYPE(TREE\_TYPE),pointer *TREE*,  
TYPE(TREE\_NODE\_TYPE),pointer *TREE\_NODE*, INTEGER(INTG),intent(in)  
*VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out)  
*ERROR*, \*)**

Sets the value at a specified tree node.

**Parameters:**

*TREE* A pointer to the tree containing the tree node  
*TREE\_NODE* A pointer to the tree node to set the value of  
*VALUE* The value of the specified tree node to set  
*ERR* The error code  
*ERROR* The error string.

Definition at line 1000 of file trees.f90.

References BASE\_ROUTINES::ENTERS(), BASE\_ROUTINES::ERRORS(), and BASE\_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.45.2.16 subroutine TREES::TREE\_OUTPUT (INTEGER(INTG),intent(in) *ID*,  
TYPE(TREE\_TYPE),pointer *TREE*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

Outputs a tree to the specified output stream ID.

**Parameters:**

- ID** The ID of the output stream
- TREE** A pointer to the tree to search
- ERR** The error code
- ERROR** The error string.

Definition at line 1038 of file trees.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    `BASE_ROUTINES::EXITS()`, and `TREE_OUTPUT_IN_ORDER()`.

Referenced by `NODE_ROUTINES::NODES_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.45.2.17 subroutine TREES::TREE\_OUTPUT\_IN\_ORDER (INTEGER(INTG),intent(in) ID, TYPE(TREE\_TYPE),pointer TREE, TYPE(TREE\_NODE\_TYPE),pointer X, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Outputs a tree in order to the specified output stream ID from the specified tree node.

**Parameters:**

- ID** The ID of the output stream
- TREE** A pointer to the tree to search
- X** A pointer to the tree node to output from
- ERR** The error code
- ERROR** The error string.

Definition at line 1074 of file trees.f90.

References    `BASE_ROUTINES::ENTERS()`,    `BASE_ROUTINES::ERRORS()`,    and    `BASE_ROUTINES::EXITS()`.

Referenced by `TREE_OUTPUT()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.45.2.18 TYPE(TREE\_NODE\_TYPE),pointer TREES::TREE\_PREDECESSOR (TYPE(TREE\_TYPE),pointer TREE, TYPE(TREE\_NODE\_TYPE),pointer X, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

Returns the predecessor of a tree at a specified tree node.

**Parameters:**

- TREE** A pointer to the Red-Black tree to find the predecessor of
- X** A pointer to the tree node to return the predecessor of

**ERR** The error code

**ERROR** The error string.

Definition at line 1129 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.45.2.19 subroutine TREES::TREE\_SEARCH (TYPE(TREE\_TYPE),pointer TREE,  
INTEGER(INTG),intent(in) KEY, TYPE(TREE\_NODE\_TYPE),pointer X,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*)**

Searches a tree to see if it contains a key.

#### Parameters:

**TREE** A pointer to the tree to search

**KEY** The key to search for

**X** On return a pointer to the tree node containing the key. If the key does not exist NULL is returned

**ERR** The error code

**ERROR** The error string.

Definition at line 1185 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `NODE_ROUTINES::NODE_CHECK_EXISTS()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.45.2.20 TYPE(TREE\_NODE\_TYPE),pointer TREES::TREE\_SUCCESSOR  
(TYPE(TREE\_TYPE),pointer TREE, TYPE(TREE\_NODE\_TYPE),pointer X,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR)  
[private]**

Returns the successor of a tree at a specified tree node.

#### Parameters:

**TREE** A pointer to the Red-Black tree to find the successor of

**X** A pointer to the tree node to return the successor of

**ERR** The error code

**ERROR** The error string.

Definition at line 1242 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by TREE\_ITEM\_DELETE().

Here is the call graph for this function:

Here is the caller graph for this function:

## 6.46 TYPES Namespace Reference

This module contains all type definitions in order to avoid cyclic module references.

### Classes

- struct [QUADRATURE\\_SCHEME\\_TYPE](#)  
*Contains information for a particular quadrature scheme.*
- struct [QUADRATURE\\_SCHEME\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [QUADRATURE\\_SCHEME\\_TYPE](#).*
- struct [QUADRATURE\\_TYPE](#)  
*Contains information on the quadrature to be used for integrating a basis.*
- struct [BASIS\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [BASIS\\_TYPE](#).*
- struct [BASIS\\_TYPE](#)  
*Contains all information about a basis .*
- struct [BASIS\\_FUNCTIONS\\_TYPE](#)  
*Contains information on the defined basis functions.*
- struct [COORDINATE\\_SYSTEM\\_TYPE](#)  
*Contains information on a coordinate system.*
- struct [NODE\\_TYPE](#)  
*Contains information about a node.*
- struct [NODES\\_TYPE](#)  
*Contains information on the nodes defined on a region.*
- struct [MESH\\_DOFS\\_TYPE](#)  
*Contains information on the dofs for a mesh.*
- struct [MESH\\_ELEMENT\\_TYPE](#)  
*Contains the information for an element in a mesh.*
- struct [MESH\\_ELEMENTS\\_TYPE](#)  
*Contains the information for the elements of a mesh.*
- struct [MESH\\_NODE\\_TYPE](#)  
*Contains the topology information for a global node of a mesh.*
- struct [MESH\\_NODES\\_TYPE](#)  
*Contains the information for the nodes of a mesh.*
- struct [MESH\\_TOPOLOGY\\_TYPE](#)

*Contains information on the (global) topology of a mesh.*

- struct [MESH\\_TOPOLOGY\\_PTR\\_TYPE](#)

*A buffer type to allow for an array of pointers to a [MESH\\_TOPOLOGY\\_TYPE](#).*

- struct [MESH\\_TYPE](#)

*Contains information on a mesh defined on a region.*

- struct [MESH\\_PTR\\_TYPE](#)

*A buffer type to allow for an array of pointers to a [MESH\\_TYPE](#).*

- struct [MESHES\\_TYPE](#)

*Contains information on the meshes defined on a region.*

- struct [GENERATED\\_MESH\\_REGULAR\\_TYPE](#)

*Contains information on a generated regular mesh.*

- struct [GENERATED\\_MESH\\_TYPE](#)

*Contains information on a generated mesh.*

- struct [GENERATED\\_MESH\\_PTR\\_TYPE](#)

*A buffer type to allow for an array of pointers to a [GENERATED\\_MESH\\_TYPE](#).*

- struct [GENERATED\\_MESHES\\_TYPE](#)

*Contains information on the generated meshes defined on a region.*

- struct [DOMAIN\\_DOFS\\_TYPE](#)

*Contains information on the degrees-of-freedom (dofs) for a domain.*

- struct [DOMAIN\\_LINE\\_TYPE](#)

*Contains the information for a line in a domain.*

- struct [DOMAIN\\_LINE\\_PTR\\_TYPE](#)

*A buffer type to allow for an array of pointers to a [DOMAIN\\_LINE\\_TYPE](#).*

- struct [DOMAIN\\_LINES\\_TYPE](#)

*Contains the topology information for the lines of a domain.*

- struct [DOMAIN\\_FACE\\_TYPE](#)

*Contains the information for a face in a domain.*

- struct [DOMAIN\\_FACE\\_PTR\\_TYPE](#)

*A buffer type to allow for an array of pointers to a [FIELD\\_VARIABLE\\_TYPE](#).*

- struct [DOMAIN\\_FACES\\_TYPE](#)

*Contains the topology information for the faces of a domain.*

- struct [DOMAIN\\_ELEMENT\\_TYPE](#)

*Contains the information for an element in a domain.*

- struct [DOMAIN\\_ELEMENTS\\_TYPE](#)  
*Contains the topology information for the elements of a domain.*
- struct [DOMAIN\\_NODE\\_TYPE](#)  
*Contains the topology information for a local node of a domain.*
- struct [DOMAIN\\_NODES\\_TYPE](#)  
*Contains the topology information for the nodes of a domain.*
- struct [DOMAIN\\_TOPOLOGY\\_TYPE](#)  
*Contains the topology information for a domain.*
- struct [DISTRIBUTED\\_VECTOR\\_TRANSFER\\_TYPE](#)  
*Contains the information for an adjacent domain for transferring the ghost data of a distributed vector to/from the current domain.*
- struct [DISTRIBUTED\\_VECTOR\\_CMISS\\_TYPE](#)  
*Contains information for a CMISS distributed vector.*
- struct [DISTRIBUTED\\_VECTOR\\_PETSC\\_TYPE](#)  
*Contains information for a PETSc distributed vector.*
- struct [DISTRIBUTED\\_VECTOR\\_TYPE](#)  
*Contains the information for a vector that is distributed across a number of domains.*
- struct [DISTRIBUTED\\_MATRIX\\_CMISS\\_TYPE](#)  
*Contains information for a CMISS distributed matrix.*
- struct [DISTRIBUTED\\_MATRIX\\_PETSC\\_TYPE](#)  
*Contains information for a PETSc distributed matrix.*
- struct [DISTRIBUTED\\_MATRIX\\_TYPE](#)  
*Contains the information for a matrix that is distributed across a number of domains.*
- struct [VECTOR\\_TYPE](#)  
*Contains information for a vector.*
- struct [MATRIX\\_TYPE](#)  
*Contains information for a matrix.*
- struct [DOMAIN\\_ADJACENT\\_DOMAIN\\_TYPE](#)  
*Contains the information on an adjacent domain to a domain in a domain mapping.*
- struct [DOMAIN\\_GLOBAL\\_MAPPING\\_TYPE](#)  
*Contains the local information for a global mapping number for a domain mapping.*
- struct [DOMAIN\\_MAPPING\\_TYPE](#)  
*Contains information on the domain mappings (i.e., local and global numberings).*
- struct [DOMAIN\\_MAPPINGS\\_TYPE](#)

*Contains information on the domain decomposition mappings.*

- struct [DOMAIN\\_TYPE](#)  
*A pointer to the domain decomposition for this domain.*
- struct [DOMAIN\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [DOMAIN\\_TYPE](#).*
- struct [DECOMPOSITION\\_LINE\\_TYPE](#)  
*Contains the information for a line in a decomposition.*
- struct [DECOMPOSITION\\_LINES\\_TYPE](#)  
*Contains the topology information for the lines of a decomposition.*
- struct [DECOMPOSITION\\_ELEMENT\\_TYPE](#)  
*Contains the information for an element in a decomposition.*
- struct [DECOMPOSITION\\_ELEMENTS\\_TYPE](#)  
*Contains the topology information for the elements of a decomposition.*
- struct [DECOMPOSITION\\_TOPOLOGY\\_TYPE](#)  
*Contains the topology information for a decomposition.*
- struct [DECOMPOSITION\\_TYPE](#)  
*Contains information on the domain decomposition.*
- struct [DECOMPOSITION\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [DECOMPOSITION\\_TYPE](#).*
- struct [DECOMPOSITIONS\\_TYPE](#)  
*Contains information on the domain decompositions defined on a mesh.*
- struct [FIELD\\_INTERPOLATED\\_POINT\\_METRICS\\_TYPE](#)  
*Contains the interpolated point coordinate metrics. Old [CMISS](#) name GL, GU, RG.*
- struct [FIELD\\_INTERPOLATED\\_POINT\\_TYPE](#)  
*Contains the interpolated value (and the derivatives wrt xi) of a field at a point. Old [CMISS](#) name XG.*
- struct [FIELD\\_INTERPOLATION\\_PARAMETERS\\_TYPE](#)  
*Contains the parameters required to interpolate a field variable within an element. Old [CMISS](#) name XE.*
- struct [FIELD\\_GEOMETRIC\\_PARAMETERS\\_TYPE](#)  
*Contains the geometric parameters (lines, faces, volumes etc.) for a geometric field decomposition.*
- struct [FIELD\\_SCALING\\_TYPE](#)  
*A type to hold the scale factors for the appropriate mesh component of a field.*
- struct [FIELD\\_SCALINGS\\_TYPE](#)  
*A type to hold the field scalings for the field.*

- struct [FIELD\\_DOF\\_TO\\_PARAM\\_MAP\\_TYPE](#)  
*A type to hold the mapping from field dof numbers to field parameters (nodes, elements, etc).*
- struct [FIELD\\_PARAM\\_TO\\_DOF\\_MAP\\_TYPE](#)  
*A type to hold the mapping from field parameters (nodes, elements, etc) to field dof numbers for a particular field variable component.*
- struct [FIELD\\_VARIABLE\\_COMPONENT\\_TYPE](#)  
*Contains information for a component of a field variable.*
- struct [FIELD\\_VARIABLE\\_TYPE](#)  
*Contains information for a field variable defined on a field.*
- struct [FIELD\\_VARIABLE\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [FIELD\\_VARIABLE\\_TYPE](#).*
- struct [FIELD\\_CREATE\\_VALUES\\_CACHE\\_TYPE](#)  
*A type to temporarily hold (cache) the user modifiable values which are used to create a field.*
- struct [FIELD\\_MAPPINGS\\_TYPE](#)  
*The type containing the mappings for the field.*
- struct [FIELD\\_PARAMETER\\_SET\\_TYPE](#)  
*A type to hold the parameter sets for a field.*
- struct [FIELD\\_PARAMETER\\_SET\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [FIELD\\_PARAMETER\\_SET\\_TYPE](#).*
- struct [FIELD\\_PARAMETER\\_SETS\\_TYPE](#)  
*A type to store the parameter sets for a field.*
- struct [FIELD\\_TYPE](#)  
*Contains information for a field defined on a region.*
- struct [FIELD\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [FIELD\\_TYPE](#).*
- struct [FIELDS\\_TYPE](#)  
*Contains information on the fields defined on a region.*
- struct [ELEMENT\\_MATRIX\\_TYPE](#)  
*Contains information for an element matrix.*
- struct [ELEMENT\\_VECTOR\\_TYPE](#)  
*Contains information for an element vector.*
- struct [EQUATIONS\\_MATRIX\\_TYPE](#)  
*Contains information about an equations matrix.*
- struct [EQUATIONS\\_MATRIX\\_PTR\\_TYPE](#)

- struct [EQUATIONS\\_JACOBIAN\\_TYPE](#)

*Contains information on the Jacobian matrix for nonlinear problems.*

- struct [EQUATIONS\\_MATRICES\\_LINEAR\\_TYPE](#)
- struct [EQUATIONS\\_MATRICES\\_NONLINEAR\\_TYPE](#)
- struct [EQUATIONS\\_MATRICES\\_RHS\\_TYPE](#)
- struct [EQUATIONS\\_MATRICES\\_SOURCE\\_TYPE](#)
- struct [EQUATIONS\\_MATRICES\\_TYPE](#)

*Contains information on the equations matrices and rhs vector.*

- struct [VARIABLE\\_TO\\_EQUATIONS\\_COLUMN\\_MAP\\_TYPE](#)

*Contains the information about the mapping of a variable DOF to an equations matrix column.*

- struct [VARIABLE\\_TO\\_EQUATIONS\\_MATRICES\\_MAP\\_TYPE](#)

*Contains the mapping for a dependent variable type to the equations matrices.*

- struct [EQUATIONS\\_MATRIX\\_TO\\_VARIABLE\\_MAP\\_TYPE](#)
- struct [EQUATIONS\\_MAPPING\\_LINEAR\\_TYPE](#)
- struct [EQUATIONS\\_JACOBIAN\\_TO\\_VARIABLE\\_MAP\\_TYPE](#)
- struct [VARIABLE\\_TO\\_EQUATIONS\\_JACOBIAN\\_MAP\\_TYPE](#)

*Contains the mapping for a dependent variable type to the nonlinear Jacobian matrix.*

- struct [EQUATIONS\\_MAPPING\\_NONLINEAR\\_TYPE](#)
- struct [EQUATIONS\\_MAPPING\\_RHS\\_TYPE](#)
- struct [EQUATIONS\\_MAPPING\\_SOURCE\\_TYPE](#)
- struct [EQUATIONS\\_MAPPING\\_CREATE\\_VALUES\\_CACHE\\_TYPE](#)
- struct [EQUATIONS\\_MAPPING\\_TYPE](#)
- struct [EQUATIONS\\_INTERPOLATION\\_TYPE](#)

*Contains information on the interpolation for the equations.*

- struct [EQUATIONS\\_LINEAR\\_DATA\\_TYPE](#)

*Contains information on any data required for a linear solution.*

- struct [EQUATIONS\\_NONLINEAR\\_DATA\\_TYPE](#)

*Contains information on any data required for a non-linear solution.*

- struct [EQUATIONS\\_TIME\\_DATA\\_TYPE](#)

*Contains information on any data required for a time-dependent equations.*

- struct [EQUATIONS\\_TYPE](#)

*Contains information about the equations in an equations set.*

- struct [EQUATIONS\\_SET\\_GEOMETRY\\_TYPE](#)

*Contains information on the geometry for an equations set.*

- struct [EQUATIONS\\_SET\\_MATERIALS\\_TYPE](#)

- struct [EQUATIONS\\_SET\\_FIXED\\_CONDITIONS\\_TYPE](#)

*Contains information on the fixed conditions for the problem.*

- struct [EQUATIONS\\_SET\\_DEPENDENT\\_TYPE](#)

*Contains information on the dependent variables for the equations set.*

- struct [EQUATIONS\\_SET\\_SOURCE\\_TYPE](#)

*Contains information on the source for the equations set.*

- struct [EQUATIONS\\_SET\\_ANALYTIC\\_TYPE](#)

*Contains information on the analytic setup for the equations set.*

- struct [EQUATIONS\\_SET\\_TYPE](#)

*Contains information on an equations set.*

- struct [EQUATIONS\\_SET\\_PTR\\_TYPE](#)

- struct [EQUATIONS\\_SETS\\_TYPE](#)

- struct [SOLVER\\_MATRIX\\_TYPE](#)

*Contains information on the solver matrix.*

- struct [SOLVER\\_MATRIX\\_PTR\\_TYPE](#)

- struct [SOLVER\\_JACOBIAN\\_TYPE](#)

*Contains information on the solver Jacobian for nonlinear problems.*

- struct [SOLVER\\_MATRICES\\_TYPE](#)

*Contains information on the solver matrices and rhs vector.*

- struct [LINEAR\\_DIRECT\\_SOLVER\\_TYPE](#)

*Contains information for a direct linear solver.*

- struct [LINEAR\\_ITERATIVE\\_SOLVER\\_TYPE](#)

*Contains information for a direct linear solver.*

- struct [LINEAR\\_SOLVER\\_TYPE](#)

*Contains information for a linear solver.*

- struct [NONLINEAR\\_LINESEARCH\\_SOLVER\\_TYPE](#)

*Contains information for a Newton line search nonlinear solver.*

- struct [NONLINEAR\\_TRUSTREGION\\_SOLVER\\_TYPE](#)

*Contains information for a Newton trust region nonlinear solver.*

- struct [NONLINEAR\\_SOLVER\\_TYPE](#)

*Contains information for a nonlinear solver.*

- struct [TIME\\_INTEGRATION\\_SOLVER\\_TYPE](#)

*Contains information for a time integration solver.*

- struct [EIGENPROBLEM\\_SOLVER\\_TYPE](#)

*Contains information for a time integration solver.*

- struct [SOLVER\\_TYPE](#)

*Contains information on the type of solver to be used.*

- struct [EQUATIONS\\_COL\\_TO\\_SOLVER\\_COLS\\_MAP\\_TYPE](#)
- struct [EQUATIONS\\_TO\\_SOLVER\\_MAPS\\_TYPE](#)
- struct [EQUATIONS\\_TO\\_SOLVER\\_MAPS\\_PTR\\_TYPE](#)

*A buffer type to allow for an array of pointers to a [EQUATIONS\\_TO\\_SOLVER\\_MAPS\\_TYPE](#).*

- struct [JACOBIAN\\_COL\\_TO\\_SOLVER\\_COLS\\_MAP\\_TYPE](#)
- struct [JACOBIAN\\_TO\\_SOLVER\\_MAP\\_TYPE](#)
- struct [VARIABLE\\_TO\\_SOLVER\\_COL\\_MAP\\_TYPE](#)

*Contains information on the mappings between field variable dofs in an equation set and the solver matrix columns (solver dofs).*

- struct [EQUATIONS\\_TO\\_SOLVER\\_MATRIX\\_MAPS\\_SM\\_TYPE](#)

*Contains information on the equations to solver matrix mappings when indexing by solver matrix number.*

- struct [EQUATIONS\\_TO\\_SOLVER\\_MATRIX\\_MAPS\\_EM\\_TYPE](#)

*Contains information on the equations to solver matrix mappings when indexing by equations matrix number.*

- struct [EQUATIONS\\_TO\\_SOLVER\\_MATRIX\\_MAPS\\_JM\\_TYPE](#)

- struct [EQUATIONS\\_ROW\\_TO\\_SOLVER\\_ROWS\\_MAP\\_TYPE](#)

*Contains information on the mapping from the equations rows in an equations set to the solver rows.*

- struct [EQUATIONS\\_SET\\_TO\\_SOLVER\\_MAP\\_TYPE](#)

*Contains information on the mappings from an equations set to the solver matrices.*

- struct [SOLVER\\_COL\\_TO\\_EQUATIONS\\_MAP\\_TYPE](#)

*Contains information about the mapping from a solver matrix column to equations matrices and variables.*

- struct [SOLVER\\_DOF\\_TO\\_VARIABLE\\_MAP\\_TYPE](#)

*Contains information about mapping the solver dof to the field variable dofs in the equations set.*

- struct [SOLVER\\_COL\\_TO\\_EQUATIONS\\_SET\\_MAP\\_TYPE](#)

*Contains information about the mappings from a solver matrix to the equations in an equations set.*

- struct [SOLVER\\_COL\\_TO\\_EQUATIONS\\_SETS\\_MAP\\_TYPE](#)

*Contains information on the mappings from a solver matrix to equations sets.*

- struct [SOLVER\\_ROW\\_TO\\_EQUATIONS\\_SET\\_MAP\\_TYPE](#)

*Contains information on the mappings from a solver row to the equations rows.*

- struct [SOLUTION\\_MAPPING\\_CREATE\\_VALUES\\_CACHE\\_TYPE](#)

*Contains information about the cached create values for a solution mapping.*

- struct [SOLUTION\\_MAPPING\\_TYPE](#)

*Contains information on the solution mapping between the global equation sets and the solver matrices.*

- struct [SOLUTION\\_TYPE](#)

*Contains information on the solution of a problem.*

- struct [SOLUTION\\_PTR\\_TYPE](#)

- struct [PROBLEM\\_LINEAR\\_DATA\\_TYPE](#)  
*Contains information on any data required for a linear solution.*
- struct [PROBLEM\\_NONLINEAR\\_DATA\\_TYPE](#)  
*Contains information on any data required for a non-linear solution.*
- struct [PROBLEM\\_TIME\\_DATA\\_TYPE](#)  
*Contains information on any data required for a time-dependent solution.*
- struct [PROBLEM\\_CONTROL\\_TYPE](#)
- struct [PROBLEM\\_TYPE](#)  
*Contains information for a problem.*
- struct [PROBLEM\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [PROBLEM\\_TYPE](#).*
- struct [PROBLEMS\\_TYPE](#)  
*Contains information on the problems defined on a region.*
- struct [REGION\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [REGION\\_TYPE](#).*
- struct [REGION\\_TYPE](#)  
*Contains information for a region.*

### 6.46.1 Detailed Description

This module contains all type definitions in order to avoid cyclic module references.

# Chapter 7

## Class Documentation

### 7.1 BASE\_ROUTINES::FLAG\_ERROR Interface Reference

Flags an error condition.

#### Private Member Functions

- subroutine [FLAG\\_ERROR\\_C](#) (STRING, ERR, ERROR,\*)
- subroutine [FLAG\\_ERROR\\_VS](#) (STRING, ERR, ERROR,\*)

#### 7.1.1 Detailed Description

Flags an error condition.

See also:

[BASE\\_ROUTINES](#)

Definition at line 190 of file base\_routines.f90.

#### 7.1.2 Member Function Documentation

##### 7.1.2.1 subroutine BASE\_ROUTINES::FLAG\_ERROR::FLAG\_ERROR\_C (CHARACTER(LEN=\*),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]

Parameters:

**STRING** The error condition string

**ERR** The error code

**ERROR** The error string

Definition at line 470 of file base\_routines.f90.

**7.1.2.2 subroutine BASE\_ROUTINES::FLAG\_ERROR::FLAG\_ERROR\_VS  
(TYPE(VARYING\_STRING),intent(in) *STRING*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

*STRING* The error condition string

*ERR* The error code

*ERROR* The error string

Definition at line 491 of file base\_routines.f90.

## 7.2 BASE\_ROUTINES::FLAG\_WARNING Interface Reference

Flags a warning to the user.

### Private Member Functions

- subroutine [FLAG\\_WARNING\\_C](#) (STRING, ERR, ERROR,\*)
- subroutine [FLAG\\_WARNING\\_VS](#) (STRING, ERR, ERROR,\*)

#### 7.2.1 Detailed Description

Flags a warning to the user.

See also:

[BASE\\_ROUTINES](#)

Definition at line 196 of file base\_routines.f90.

#### 7.2.2 Member Function Documentation

**7.2.2.1 subroutine BASE\_ROUTINES::FLAG\_WARNING::FLAG\_WARNING\_C  
(CHARACTER(LEN=\*),intent(in) STRING, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Parameters:

**STRING** The warning string

**ERR** The error code

**ERROR** The error string

Definition at line 510 of file base\_routines.f90.

**7.2.2.2 subroutine BASE\_ROUTINES::FLAG\_WARNING::FLAG\_WARNING\_VS  
(TYPE(VARYING\_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Parameters:

**STRING** The warning string

**ERR** The error code

**ERROR** The error string

Definition at line 531 of file base\_routines.f90.

## 7.3 BASE\_ROUTINES::interface Interface Reference

### Private Member Functions

- subroutine **CPUTIMER** (RETURN\_TIME, TIME\_TYPE, ERR, CERROR)

#### 7.3.1 Detailed Description

Definition at line 173 of file base\_routines.f90.

#### 7.3.2 Member Function Documentation

**7.3.2.1 subroutine BASE\_ROUTINES::interface::CPUTIMER (REAL(DP),intent(out) RETURN\_TIME, INTEGER(INTG),intent(in) TIME\_TYPE, INTEGER(INTG),intent(out) ERR, INTEGER(INTG),dimension(\*),intent(out) CERROR) [private]**

Definition at line 178 of file base\_routines.f90.

## 7.4 BASE\_ROUTINES::ROUTINE\_LIST\_ITEM\_TYPE Struct Reference

Contains information for an item in the routine list for diagnostics or timing.

Collaboration diagram for BASE\_ROUTINES::ROUTINE\_LIST\_ITEM\_TYPE:

### Private Attributes

- TYPE(VARYING\_STRING) NAME  
*Name of the routine.*
- INTEGER(INTG) NUMBER\_OF\_INVOCATIONS  
*Number of times the routine has been invoked.*
- REAL(SP) TOTAL\_INCLUSIVE\_CPU\_TIME  
*Total User CPU time spent in the routine inclusive of calls.*
- REAL(SP) TOTAL\_INCLUSIVE\_SYSTEM\_TIME  
*Total System CPU time spent in the routine inclusive of calls.*
- REAL(SP) TOTAL\_EXCLUSIVE\_CPU\_TIME  
*Total User CPU time spent in the routine exclusive of calls.*
- REAL(SP) TOTAL\_EXCLUSIVE\_SYSTEM\_TIME  
*Total System CPU time spent in the routine exclusive of calls.*
- TYPE(ROUTINE\_LIST\_ITEM\_TYPE), pointer NEXT\_ROUTINE  
*Pointer to the next routine item in the routine list.*

### 7.4.1 Detailed Description

Contains information for an item in the routine list for diagnostics or timing.

Definition at line 110 of file base\_routines.f90.

### 7.4.2 Member Data Documentation

#### 7.4.2.1 TYPE(VARYING\_STRING) BASE\_ROUTINES::ROUTINE\_LIST\_ITEM\_TYPE::NAME [private]

Name of the routine.

Definition at line 111 of file base\_routines.f90.

**7.4.2.2 TYPE(ROUTINE\_LIST\_ITEM\_TYPE),pointer BASE\_ROUTINES::ROUTINE\_LIST\_-  
ITEM\_TYPE::NEXT\_ROUTINE [private]**

Pointer to the next routine item in the routine list.

Definition at line 117 of file base\_routines.f90.

**7.4.2.3 INTEGER(INTG) BASE\_ROUTINES::ROUTINE\_LIST\_ITEM\_TYPE::NUMBER\_OF\_-  
INVOCATIONS [private]**

Number of times the routine has been invoked.

Definition at line 112 of file base\_routines.f90.

**7.4.2.4 REAL(SP) BASE\_ROUTINES::ROUTINE\_LIST\_ITEM\_TYPE::TOTAL\_-  
EXCLUSIVE\_CPU\_TIME [private]**

Total User CPU time spent in the routine exclusive of calls.

Definition at line 115 of file base\_routines.f90.

**7.4.2.5 REAL(SP) BASE\_ROUTINES::ROUTINE\_LIST\_ITEM\_TYPE::TOTAL\_-  
EXCLUSIVE\_SYSTEM\_TIME [private]**

Total System CPU time spent in the routine exclusive of calls.

Definition at line 116 of file base\_routines.f90.

**7.4.2.6 REAL(SP) BASE\_ROUTINES::ROUTINE\_LIST\_ITEM\_TYPE::TOTAL\_INCLUSIVE\_-  
CPU\_TIME [private]**

Total User CPU time spent in the routine inclusive of calls.

Definition at line 113 of file base\_routines.f90.

**7.4.2.7 REAL(SP) BASE\_ROUTINES::ROUTINE\_LIST\_ITEM\_TYPE::TOTAL\_INCLUSIVE\_-  
SYSTEM\_TIME [private]**

Total System CPU time spent in the routine inclusive of calls.

Definition at line 114 of file base\_routines.f90.

## 7.5 BASE\_ROUTINES::ROUTINE\_LIST\_TYPE Struct Reference

Contains information for the routine list for diagnostics or timing.

Collaboration diagram for BASE\_ROUTINES::ROUTINE\_LIST\_TYPE:

### Private Attributes

- TYPE(ROUTINE\_LIST\_ITEM\_TYPE), pointer HEAD  
*A pointer to the head of the routine list.*

#### 7.5.1 Detailed Description

Contains information for the routine list for diagnostics or timing.

Definition at line 121 of file base\_routines.f90.

#### 7.5.2 Member Data Documentation

##### 7.5.2.1 TYPE(ROUTINE\_LIST\_ITEM\_TYPE),pointer BASE\_ROUTINES::ROUTINE\_LIST\_- TYPE::HEAD [private]

A pointer to the head of the routine list.

Definition at line 122 of file base\_routines.f90.

## 7.6 BASE\_ROUTINES::ROUTINE\_STACK\_ITEM\_TYPE Struct Reference

Contains information for an item in the routine invocation stack.

Collaboration diagram for BASE\_ROUTINES::ROUTINE\_STACK\_ITEM\_TYPE:

### Private Attributes

- TYPE(VARYING\_STRING) NAME  
*The name of the routine.*
- REAL(SP) INCLUSIVE\_CPU\_TIME  
*User CPU time spent in the routine inclusive of calls.*
- REAL(SP) INCLUSIVE\_SYSTEM\_TIME  
*System CPU time spent in the routine inclusive of calls.*
- REAL(SP) EXCLUSIVE\_CPU\_TIME  
*User CPU time spent in the routine exclusive of calls.*
- REAL(SP) EXCLUSIVE\_SYSTEM\_TIME  
*System CPU time spent in the routine exclusive of calls.*
- LOGICAL DIAGNOSTICS  
*.TRUE. if diagnostics are active in the routine*
- LOGICAL TIMING  
*.TRUE. if timing is active in the routine*
- TYPE(ROUTINE\_LIST\_ITEM\_TYPE), pointer ROUTINE\_LIST\_ITEM  
*Pointer to the routine list item for diagnostics or timing.*
- TYPE(ROUTINE\_STACK\_ITEM\_TYPE), pointer PREVIOUS\_ROUTINE  
*Pointer to the previous routine in the routine stack.*

### 7.6.1 Detailed Description

Contains information for an item in the routine invocation stack.

Definition at line 126 of file base\_routines.f90.

### 7.6.2 Member Data Documentation

#### 7.6.2.1 LOGICAL BASE\_ROUTINES::ROUTINE\_STACK\_ITEM\_TYPE::DIAGNOSTICS [private]

.TRUE. if diagnostics are active in the routine

Definition at line 132 of file base\_routines.f90.

**7.6.2.2 REAL(SP) BASE\_ROUTINES::ROUTINE\_STACK\_ITEM\_TYPE::EXCLUSIVE\_CPU\_-  
TIME [private]**

User CPU time spent in the routine exclusive of calls.

Definition at line 130 of file base\_routines.f90.

**7.6.2.3 REAL(SP) BASE\_ROUTINES::ROUTINE\_STACK\_ITEM\_TYPE::EXCLUSIVE\_-  
SYSTEM\_TIME [private]**

System CPU time spent in the routine exclusive of calls.

Definition at line 131 of file base\_routines.f90.

**7.6.2.4 REAL(SP) BASE\_ROUTINES::ROUTINE\_STACK\_ITEM\_TYPE::INCLUSIVE\_CPU\_-  
TIME [private]**

User CPU time spent in the routine inclusive of calls.

Definition at line 128 of file base\_routines.f90.

**7.6.2.5 REAL(SP) BASE\_ROUTINES::ROUTINE\_STACK\_ITEM\_TYPE::INCLUSIVE\_-  
SYSTEM\_TIME [private]**

System CPU time spent in the routine inclusive of calls.

Definition at line 129 of file base\_routines.f90.

**7.6.2.6 TYPE(VARYING\_STRING) BASE\_ROUTINES::ROUTINE\_STACK\_ITEM\_-  
TYPE::NAME [private]**

The name of the routine.

Definition at line 127 of file base\_routines.f90.

**7.6.2.7 TYPE(ROUTINE\_STACK\_ITEM\_TYPE),pointer BASE\_ROUTINES::ROUTINE\_-  
STACK\_ITEM\_TYPE::PREVIOUS\_ROUTINE [private]**

Pointer to the previous routine in the routine stack.

Definition at line 135 of file base\_routines.f90.

**7.6.2.8 TYPE(ROUTINE\_LIST\_ITEM\_TYPE),pointer BASE\_ROUTINES::ROUTINE\_-  
STACK\_ITEM\_TYPE::ROUTINE\_LIST\_ITEM [private]**

Pointer to the routine list item for diagnostics or timing.

Definition at line 134 of file base\_routines.f90.

**7.6.2.9 LOGICAL BASE\_ROUTINES::ROUTINE\_STACK\_ITEM\_TYPE::TIMING  
[private]**

.TRUE. if timing is active in the routine

Definition at line 133 of file base\_routines.f90.

## 7.7 BASE\_ROUTINES::ROUTINE\_STACK\_TYPE Struct Reference

Contains information for the routine invocation stack.

Collaboration diagram for BASE\_ROUTINES::ROUTINE\_STACK\_TYPE:

### Private Attributes

- TYPE(ROUTINE\_STACK\_ITEM\_TYPE), pointer STACK\_POINTER  
*Pointer to the top of the stack.*

#### 7.7.1 Detailed Description

Contains information for the routine invocation stack.

Definition at line 139 of file base\_routines.f90.

#### 7.7.2 Member Data Documentation

##### 7.7.2.1 TYPE(ROUTINE\_STACK\_ITEM\_TYPE),pointer BASE\_ROUTINES::ROUTINE\_STACK\_TYPE::STACK\_POINTER [private]

Pointer to the top of the stack.

Definition at line 140 of file base\_routines.f90.

## 7.8 BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE Struct Reference

### Public Attributes

- INTEGER(INTG) FILE\_NUMBER
- INTEGER(INTG) BINARY\_FILE\_REVISION
- INTEGER(INTG) MACHINE\_TYPE
- INTEGER(INTG) OS\_TYPE
- INTEGER(INTG) ENDIAN\_TYPE
- INTEGER(INTG) CHAR\_FORMAT
- INTEGER(INTG) INT\_FORMAT
- INTEGER(INTG) SP\_FORMAT
- INTEGER(INTG) DP\_FORMAT
- INTEGER(INTG) CHARACTER\_SIZE
- INTEGER(INTG) INTEGER\_SIZE
- INTEGER(INTG) SINTEGER\_SIZE
- INTEGER(INTG) LINTEGER\_SIZE
- INTEGER(INTG) SP\_REAL\_SIZE
- INTEGER(INTG) DP\_REAL\_SIZE
- INTEGER(INTG) LOGICAL\_SIZE
- INTEGER(INTG) SPC\_REAL\_SIZE
- INTEGER(INTG) DPC\_REAL\_SIZE
- CHARACTER(LEN=MAXSTRLEN) FILE\_NAME
- INTEGER(INTG) ACCESS\_TYPE

### 7.8.1 Detailed Description

Definition at line 263 of file binary\_file\_f.f90.

### 7.8.2 Member Data Documentation

#### 7.8.2.1 INTEGER(INTG) BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE::ACCESS\_TYPE

Definition at line 284 of file binary\_file\_f.f90.

#### 7.8.2.2 INTEGER(INTG) BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE::BINARY\_FILE\_REVISION

Definition at line 266 of file binary\_file\_f.f90.

#### 7.8.2.3 INTEGER(INTG) BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE::CHAR\_FORMAT

Definition at line 270 of file binary\_file\_f.f90.

#### 7.8.2.4 INTEGER(INTG) BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE::CHARACTER\_SIZE

Definition at line 274 of file binary\_file\_f.f90.

**7.8.2.5 INTEGER(INTG) BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE::DP\_FORMAT**

Definition at line 273 of file binary\_file\_f.f90.

**7.8.2.6 INTEGER(INTG) BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE::DP\_REAL\_SIZE**

Definition at line 279 of file binary\_file\_f.f90.

**7.8.2.7 INTEGER(INTG) BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE::DPC\_REAL\_SIZE**

Definition at line 282 of file binary\_file\_f.f90.

**7.8.2.8 INTEGER(INTG) BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE::ENDIAN\_TYPE**

Definition at line 269 of file binary\_file\_f.f90.

**7.8.2.9 CHARACTER(LEN=MAXSTRLEN) BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE::FILE\_NAME**

Definition at line 283 of file binary\_file\_f.f90.

**7.8.2.10 INTEGER(INTG) BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE::FILE\_NUMBER**

Definition at line 265 of file binary\_file\_f.f90.

**7.8.2.11 INTEGER(INTG) BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE::INT\_FORMAT**

Definition at line 271 of file binary\_file\_f.f90.

**7.8.2.12 INTEGER(INTG) BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE::INTEGER\_SIZE**

Definition at line 275 of file binary\_file\_f.f90.

**7.8.2.13 INTEGER(INTG) BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE::LINTEGER\_SIZE**

Definition at line 277 of file binary\_file\_f.f90.

**7.8.2.14 INTEGER(INTG) BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE::LOGICAL\_SIZE**

Definition at line 280 of file binary\_file\_f.f90.

**7.8.2.15 INTEGER(INTG) BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE::MACHINE\_TYPE**

Definition at line 267 of file binary\_file\_f.f90.

**7.8.2.16 INTEGER(INTG) BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE::OS\_TYPE**

Definition at line 268 of file binary\_file\_f.f90.

**7.8.2.17 INTEGER(INTG) BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE::SINTEGER\_SIZE**

Definition at line 276 of file binary\_file\_f.f90.

**7.8.2.18 INTEGER(INTG) BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE::SP\_FORMAT**

Definition at line 272 of file binary\_file\_f.f90.

**7.8.2.19 INTEGER(INTG) BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE::SP\_REAL\_SIZE**

Definition at line 278 of file binary\_file\_f.f90.

**7.8.2.20 INTEGER(INTG) BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE::SPC\_REAL\_SIZE**

Definition at line 281 of file binary\_file\_f.f90.

## 7.9 BINARY\_FILE::BINARY\_FILE\_TYPE Struct Reference

Collaboration diagram for BINARY\_FILE::BINARY\_FILE\_TYPE:

### Public Attributes

- TYPE(BINARY\_FILE\_INFO\_TYPE), pointer FILE\_INFORMATION

#### 7.9.1 Detailed Description

Definition at line 287 of file binary\_file\_f.f90.

#### 7.9.2 Member Data Documentation

##### 7.9.2.1 TYPE(BINARY\_FILE\_INFO\_TYPE),pointer BINARY\_FILE::BINARY\_FILE\_TYPE::FILE\_INFORMATION

Definition at line 288 of file binary\_file\_f.f90.

## 7.10 BINARY\_FILE::BINARY\_TAG\_TYPE Struct Reference

### Public Attributes

- INTEGER(INTG) [INDEX](#)
- INTEGER(INTG) [NUM\\_SUBTAGS](#)
- INTEGER(INTG) [NUM\\_BYTES](#)
- INTEGER(INTG) [NUM\\_HEADER\\_BYTES](#)
- CHARACTER(LEN=MAXSTRLEN) [HEADER](#)

#### 7.10.1 Detailed Description

Definition at line 291 of file binary\_file.f90.

#### 7.10.2 Member Data Documentation

##### 7.10.2.1 CHARACTER(LEN=MAXSTRLEN) BINARY\_FILE::BINARY\_TAG\_TYPE::HEADER

Definition at line 296 of file binary\_file.f90.

##### 7.10.2.2 INTEGER(INTG) BINARY\_FILE::BINARY\_TAG\_TYPE::INDEX

Definition at line 292 of file binary\_file.f90.

##### 7.10.2.3 INTEGER(INTG) BINARY\_FILE::BINARY\_TAG\_TYPE::NUM\_BYTES

Definition at line 294 of file binary\_file.f90.

##### 7.10.2.4 INTEGER(INTG) BINARY\_FILE::BINARY\_TAG\_TYPE::NUM\_HEADER\_BYTES

Definition at line 295 of file binary\_file.f90.

##### 7.10.2.5 INTEGER(INTG) BINARY\_FILE::BINARY\_TAG\_TYPE::NUM\_SUBTAGS

Definition at line 293 of file binary\_file.f90.

## 7.11 BINARY\_FILE::interface Interface Reference

### Public Member Functions

- subroutine **BINARYCLOSEFILE** (FILE\_NUMBER, ERR, CERROR)
- subroutine **BINARYOPENFILE** (FILE\_NUMBER, CFNAME, CACCESSCODE, ERR, CERROR)
- subroutine **BINARYSETFILE** (FILE\_NUMBER, SET\_CODE, ERR, CERROR)
- subroutine **BINARYSKIPFILE** (FILE\_NUMBER, NUMBER\_BYTES, ERR, CERROR)
- subroutine **ISBINARYFILEOPEN** (FILE\_NUMBER, RETURNCODE, ERR, CERROR)
- subroutine **ISENDBINARYFILE** (FILE\_NUMBER, RETURN\_CODE, ERR, CERROR)

#### 7.11.1 Detailed Description

Definition at line 305 of file binary\_file.f90.

#### 7.11.2 Member Function Documentation

**7.11.2.1 subroutine BINARY\_FILE::interface::BINARYCLOSEFILE  
(INTEGER(INTG),intent(in) FILE\_NUMBER, INTEGER(INTG),intent(out) ERR,  
INTEGER(INTG),dimension(\*),intent(out) CERROR)**

Definition at line 307 of file binary\_file.f90.

**7.11.2.2 subroutine BINARY\_FILE::interface::BINARYOPENFILE (INTEGER(INTG),intent(in)  
FILE\_NUMBER, INTEGER(INTG),dimension(\*),intent(in)  
CFNAME, INTEGER(INTG),dimension(\*),intent(in) CACCESSCODE,  
INTEGER(INTG),intent(out) ERR, INTEGER(INTG),dimension(\*),intent(out)  
CERROR)**

Definition at line 314 of file binary\_file.f90.

**7.11.2.3 subroutine BINARY\_FILE::interface::BINARYSETFILE (INTEGER(INTG),intent(in)  
FILE\_NUMBER, INTEGER(INTG),intent(in) SET\_CODE,  
INTEGER(INTG),intent(out) ERR, INTEGER(INTG),dimension(\*),intent(out)  
CERROR)**

Definition at line 323 of file binary\_file.f90.

**7.11.2.4 subroutine BINARY\_FILE::interface::BINARYSKIPFILE (INTEGER(INTG),intent(in)  
FILE\_NUMBER, INTEGER(INTG),intent(in) NUMBER\_BYTES,  
INTEGER(INTG),intent(out) ERR, INTEGER(INTG),dimension(\*),intent(out)  
CERROR)**

Definition at line 330 of file binary\_file.f90.

**7.11.2.5 subroutine `BINARY_FILE::interface::ISBINARYFILEOPEN`** (INTEGER(INTG),intent(in) *FILE\_NUMBER*, INTEGER(INTG),intent(out) *RETURNCODE*, INTEGER(INTG),intent(out) *ERR*, INTEGER(INTG),dimension(\*),intent(out) *CERROR*)

Definition at line 341 of file binary\_file\_f.f90.

**7.11.2.6 subroutine `BINARY_FILE::interface::ISENDMEMORYFILE`** (INTEGER(INTG),intent(in) *FILE\_NUMBER*, INTEGER(INTG),intent(out) *RETURN\_CODE*, INTEGER(INTG),intent(out) *ERR*, INTEGER(INTG),dimension(\*),intent(out) *CERROR*)

Definition at line 348 of file binary\_file\_f.f90.

## 7.12 BINARY\_FILE::READ\_BINARY\_FILE Interface Reference

### Public Member Functions

- subroutine [READ\\_BINARY\\_FILE\\_INTG](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [READ\\_BINARY\\_FILE\\_INTG1](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [READ\\_BINARY\\_FILE\\_SINTG](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [READ\\_BINARY\\_FILE\\_SINTG1](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [READ\\_BINARY\\_FILE\\_LINTG](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [READ\\_BINARY\\_FILE\\_LINTG1](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [READ\\_BINARY\\_FILE\\_SP](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [READ\\_BINARY\\_FILE\\_SP1](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [READ\\_BINARY\\_FILE\\_DP](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [READ\\_BINARY\\_FILE\\_DP1](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [READ\\_BINARY\\_FILE\\_CHARACTER](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [READ\\_BINARY\\_FILE\\_LOGICAL](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [READ\\_BINARY\\_FILE\\_LOGICAL1](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [READ\\_BINARY\\_FILE\\_SPC](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [READ\\_BINARY\\_FILE\\_SPC1](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [READ\\_BINARY\\_FILE\\_DPC](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [READ\\_BINARY\\_FILE\\_DPC1](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)

#### 7.12.1 Detailed Description

Definition at line 357 of file binary\_file\_f.f90.

#### 7.12.2 Member Function Documentation

**7.12.2.1 subroutine BINARY\_FILE::READ\_BINARY\_FILE::READ\_BINARY\_-  
FILE\_CHARACTER (TYPE(BINARY\_FILE\_TYPE),intent(in) FILEID,  
INTEGER(INTG),intent(in) NUM\_DATA, CHARACTER(LEN=\*),intent(out) DATA,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Definition at line 1341 of file binary\_file\_f.f90.

**7.12.2.2 subroutine BINARY\_FILE::READ\_BINARY\_FILE::READ\_BINARY\_FILE\_DP  
(TYPE(BINARY\_FILE\_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)  
NUM\_DATA, REAL(DP),dimension(\*),intent(out) DATA, INTEGER(INTG),intent(out)  
ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Definition at line 1243 of file binary\_file\_f.f90.

**7.12.2.3 subroutine BINARY\_FILE::READ\_BINARY\_FILE::READ\_BINARY\_FILE\_DP1  
(TYPE(BINARY\_FILE\_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)  
NUM\_DATA, REAL(DP),intent(out) DATA, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Definition at line 1290 of file binary\_file\_f.f90.

---

**7.12.2.4 subroutine `BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_DPC`**  
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)`  
`NUM_DATA, COMPLEX(DPC),dimension(*),intent(out) DATA,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 1591 of file binary\_file\_f.f90.

**7.12.2.5 subroutine `BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_DPC1`**  
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)`  
`NUM_DATA, COMPLEX(DPC),intent(out) DATA, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 1639 of file binary\_file\_f.f90.

**7.12.2.6 subroutine `BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_INTG`**  
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)`  
`NUM_DATA, INTEGER(INTG),dimension(*),intent(out) DATA,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 851 of file binary\_file\_f.f90.

**7.12.2.7 subroutine `BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_INTG1`**  
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)`  
`NUM_DATA, INTEGER(INTG),intent(out) DATA, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 898 of file binary\_file\_f.f90.

**7.12.2.8 subroutine `BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_LINTG`**  
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)`  
`NUM_DATA, INTEGER(LINTG),dimension(*),intent(out) DATA,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 1047 of file binary\_file\_f.f90.

**7.12.2.9 subroutine `BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_LINTG1`**  
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)`  
`NUM_DATA, INTEGER(LINTG),intent(out) DATA, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 1094 of file binary\_file\_f.f90.

**7.12.2.10 subroutine `BINARY_FILE::READ_BINARY_FILE::READ_BINARY_-`**  
`FILE_LOGICAL (TYPE(BINARY_FILE_TYPE),intent(in) FILEID,`  
`INTEGER(INTG),intent(in) NUM_DATA, LOGICAL,dimension(*),intent(out) DATA,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`  
`*)`

Definition at line 1393 of file binary\_file\_f.f90.

---

7.12.2.11 subroutine **BINARY\_FILE::READ\_BINARY\_FILE::READ\_BINARY\_-  
FILE\_LOGICAL1** (TYPE(BINARY\_FILE\_TYPE),intent(in) *FILEID*,  
INTEGER(INTG),intent(in) *NUM\_DATA*, LOGICAL,intent(out) *DATA*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
\*)

Definition at line 1440 of file binary\_file\_f.f90.

7.12.2.12 subroutine **BINARY\_FILE::READ\_BINARY\_FILE::READ\_BINARY\_FILE\_SINTG**  
(TYPE(BINARY\_FILE\_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in)  
*NUM\_DATA*, INTEGER(SINTG),dimension(\*),intent(out) *DATA*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
\*)

Definition at line 949 of file binary\_file\_f.f90.

7.12.2.13 subroutine **BINARY\_FILE::READ\_BINARY\_FILE::READ\_BINARY\_FILE\_SINTG1**  
(TYPE(BINARY\_FILE\_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in)  
*NUM\_DATA*, INTEGER(SINTG),intent(out) *DATA*, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Definition at line 996 of file binary\_file\_f.f90.

7.12.2.14 subroutine **BINARY\_FILE::READ\_BINARY\_FILE::READ\_BINARY\_FILE\_SP**  
(TYPE(BINARY\_FILE\_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in)  
*NUM\_DATA*, REAL(SP),dimension(\*),intent(out) *DATA*, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Definition at line 1145 of file binary\_file\_f.f90.

7.12.2.15 subroutine **BINARY\_FILE::READ\_BINARY\_FILE::READ\_BINARY\_FILE\_SP1**  
(TYPE(BINARY\_FILE\_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in)  
*NUM\_DATA*, REAL(SP),intent(out) *DATA*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

Definition at line 1192 of file binary\_file\_f.f90.

7.12.2.16 subroutine **BINARY\_FILE::READ\_BINARY\_FILE::READ\_BINARY\_FILE\_SPC**  
(TYPE(BINARY\_FILE\_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in)  
*NUM\_DATA*, COMPLEX(SPC),dimension(\*),intent(out) *DATA*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
\*)

Definition at line 1491 of file binary\_file\_f.f90.

**7.12.2.17 subroutine `BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_SPC1`**  
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)`  
`NUM_DATA, COMPLEX(SPC),intent(out) DATA, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 1539 of file binary\_file\_f.f90.

## 7.13 BINARY\_FILE::WRITE\_BINARY\_FILE Interface Reference

### Public Member Functions

- subroutine `WRITE_BINARY_FILE_INTG` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `WRITE_BINARY_FILE_INTG1` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `WRITE_BINARY_FILE_SINTG` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `WRITE_BINARY_FILE_SINTG1` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `WRITE_BINARY_FILE_LINTG` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `WRITE_BINARY_FILE_LINTG1` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `WRITE_BINARY_FILE_SP` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `WRITE_BINARY_FILE_SP1` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `WRITE_BINARY_FILE_DP` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `WRITE_BINARY_FILE_DP1` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `WRITE_BINARY_FILE_CHARACTER` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `WRITE_BINARY_FILE_LOGICAL` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `WRITE_BINARY_FILE_LOGICAL1` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `WRITE_BINARY_FILE_SPC` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `WRITE_BINARY_FILE_SPC1` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `WRITE_BINARY_FILE_DPC` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `WRITE_BINARY_FILE_DPC1` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)

#### 7.13.1 Detailed Description

Definition at line 377 of file binary\_file\_f.f90.

#### 7.13.2 Member Function Documentation

**7.13.2.1 subroutine BINARY\_FILE::WRITE\_BINARY\_FILE::WRITE\_BINARY\_-  
FILE\_CHARACTER (TYPE(BINARY\_FILE\_TYPE),intent(in) FILEID,  
INTEGER(INTG),intent(in) NUM\_DATA, CHARACTER(LEN=\*),intent(in) DATA,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Definition at line 2466 of file binary\_file\_f.f90.

**7.13.2.2 subroutine BINARY\_FILE::WRITE\_BINARY\_FILE::WRITE\_BINARY\_FILE\_DP  
(TYPE(BINARY\_FILE\_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)  
NUM\_DATA, REAL(DP),dimension(\*),intent(in) DATA, INTEGER(INTG),intent(out)  
ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Definition at line 2376 of file binary\_file\_f.f90.

**7.13.2.3 subroutine BINARY\_FILE::WRITE\_BINARY\_FILE::WRITE\_BINARY\_FILE\_DP1  
(TYPE(BINARY\_FILE\_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)  
NUM\_DATA, REAL(DP),intent(in) DATA, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Definition at line 2420 of file binary\_file\_f.f90.

---

**7.13.2.4 subroutine `BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_FILE_DPC`**  
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_-`  
`DATA, COMPLEX(DPC),dimension(*),intent(in) DATA, INTEGER(INTG),intent(out)`  
`ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 2689 of file binary\_file\_f.f90.

**7.13.2.5 subroutine `BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_FILE_DPC1`**  
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_-`  
`DATA, COMPLEX(DPC),intent(in) DATA, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 2733 of file binary\_file\_f.f90.

**7.13.2.6 subroutine `BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_FILE_INTG`**  
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_-`  
`DATA, INTEGER(INTG),dimension(*),intent(in) DATA, INTEGER(INTG),intent(out)`  
`ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 2022 of file binary\_file\_f.f90.

**7.13.2.7 subroutine `BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_FILE_INTG1`**  
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_-`  
`DATA, INTEGER(INTG),intent(in) DATA, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 2063 of file binary\_file\_f.f90.

**7.13.2.8 subroutine `BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_FILE_LINTG`**  
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_-`  
`DATA, INTEGER(LINTG),dimension(*),intent(in) DATA,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 2197 of file binary\_file\_f.f90.

**7.13.2.9 subroutine `BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_FILE_-`**  
**LINTG1** `(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_-`  
`DATA, INTEGER(LINTG),intent(in) DATA, INTEGER(INTG),intent(out) ERR,`  
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 2240 of file binary\_file\_f.f90.

**7.13.2.10 subroutine `BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_-`**  
**FILE\_LOGICAL** `(TYPE(BINARY_FILE_TYPE),intent(in) FILEID,`  
`INTEGER(INTG),intent(in) NUM_-DATA, LOGICAL,dimension(*),intent(in) DATA,`  
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`  
`*)`

Definition at line 2510 of file binary\_file\_f.f90.

**7.13.2.11 subroutine BINARY\_FILE::WRITE\_BINARY\_FILE::WRITE\_BINARY\_-  
FILE\_LOGICAL1 (TYPE(BINARY\_FILE\_TYPE),intent(in) FILEID,  
INTEGER(INTG),intent(in) NUM\_DATA, LOGICAL,intent(in) DATA,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*)**

Definition at line 2552 of file binary\_file\_f90.

**7.13.2.12 subroutine BINARY\_FILE::WRITE\_BINARY\_FILE::WRITE\_BINARY\_-  
FILE\_SINTG (TYPE(BINARY\_FILE\_TYPE),intent(in) FILEID,  
INTEGER(INTG),intent(in) NUM\_DATA, INTEGER(SINTG),dimension(\*),intent(in)  
DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out)  
ERROR, \*)**

Definition at line 2108 of file binary\_file\_f90.

**7.13.2.13 subroutine BINARY\_FILE::WRITE\_BINARY\_FILE::WRITE\_BINARY\_-  
FILE\_SINTG1 (TYPE(BINARY\_FILE\_TYPE),intent(in) FILEID,  
INTEGER(INTG),intent(in) NUM\_DATA, INTEGER(SINTG),intent(in) DATA,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*)**

Definition at line 2151 of file binary\_file\_f90.

**7.13.2.14 subroutine BINARY\_FILE::WRITE\_BINARY\_FILE::WRITE\_BINARY\_FILE\_SP  
(TYPE(BINARY\_FILE\_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)  
NUM\_DATA, REAL(SP),dimension(\*),intent(in) DATA, INTEGER(INTG),intent(out)  
ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Definition at line 2286 of file binary\_file\_f90.

**7.13.2.15 subroutine BINARY\_FILE::WRITE\_BINARY\_FILE::WRITE\_BINARY\_FILE\_SP1  
(TYPE(BINARY\_FILE\_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)  
NUM\_DATA, REAL(SP),intent(in) DATA, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

Definition at line 2330 of file binary\_file\_f90.

**7.13.2.16 subroutine BINARY\_FILE::WRITE\_BINARY\_FILE::WRITE\_BINARY\_FILE\_SPC  
(TYPE(BINARY\_FILE\_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)  
NUM\_DATA, COMPLEX(SPC),dimension(\*),intent(in) DATA,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*)**

Definition at line 2598 of file binary\_file\_f90.

7.13.2.17 subroutine **BINARY\_FILE::WRITE\_BINARY\_FILE::WRITE\_BINARY\_FILE\_SPC1**  
(**TYPE(BINARY\_FILE\_TYPE),intent(in)** *FILEID*, **INTEGER(INTG),intent(in)**  
*NUM\_DATA*, **COMPLEX(SPC),intent(in)** *DATA*, **INTEGER(INTG),intent(out)** *ERR*,  
**TYPE(VARYING\_STRING),intent(out)** *ERROR*, \*)

Definition at line 2642 of file binary\_file\_f.f90.

## 7.14 BLAS::interface Interface Reference

### Public Member Functions

- subroutine [SASUM](#) (N, X, INCX)
- subroutine [DASUM](#) (N, X, INCX)
- subroutine [SAXPY](#) (N, A, X, INCX, Y, INCY)
- subroutine [DAXPY](#) (N, A, X, INCX, Y, INCY)
- subroutine [SCOPY](#) (N, DX, INCX, DY, INCY)
- subroutine [DCOPY](#) (N, DX, INCX, DY, INCY)
- REAL(SP) [SDOT](#) (N, X, INCX, Y, INCY)
- REAL(DP) [DDOT](#) (N, X, INCX, Y, INCY)
- REAL(SP) [SNRM2](#) (N, X, INCX)
- REAL(DP) [DNRM2](#) (N, X, INCX)
- subroutine [SROT](#) (N, DX, INCX, DY, INCY, C, S)
- subroutine [DROT](#) (N, DX, INCX, DY, INCY, C, S)
- subroutine [SROTG](#) (DA, DB, C, S)
- subroutine [DROTG](#) (DA, DB, C, S)
- subroutine [SSCAL](#) (N, A, X, INCX)
- subroutine [DSCAL](#) (N, A, X, INCX)
- subroutine [STRSV](#) (UPLO, TRANS, DIAG, N, A, LDA, X, INCX)
- subroutine [DTRSV](#) (UPLO, TRANS, DIAG, N, A, LDA, X, INCX)

### 7.14.1 Detailed Description

Definition at line 50 of file blas.f90.

### 7.14.2 Member Function Documentation

#### 7.14.2.1 subroutine BLAS::interface::DASUM (INTEGER(INTG),intent(in) *N*, REAL(DP),dimension(\*),intent(in) *X*, INTEGER(INTG),intent(in) *INCX*)

Definition at line 61 of file blas.f90.

#### 7.14.2.2 subroutine BLAS::interface::DAXPY (INTEGER(INTG),intent(in) *N*, REAL(DP),intent(in) *A*, REAL(DP),dimension(\*),intent(in) *X*, INTEGER(INTG),intent(in) *INCX*, REAL(DP),dimension(\*),intent(out) *Y*, INTEGER(INTG),intent(in) *INCY*)

Definition at line 78 of file blas.f90.

#### 7.14.2.3 subroutine BLAS::interface::DCOPY (INTEGER(INTG),intent(in) *N*, REAL(DP),dimension(\*),intent(in) *DX*, INTEGER(INTG),intent(in) *INCX*, REAL(DP),dimension(\*),intent(out) *DY*, INTEGER(INTG),intent(in) *INCY*)

Definition at line 97 of file blas.f90.

---

**7.14.2.4 REAL(DP) BLAS::interface::DDOT (INTEGER(INTG),intent(in) *N*,  
REAL(DP),dimension(\*),intent(in) *X*, INTEGER(INTG),intent(in) *INCX*,  
REAL(DP),dimension(\*),intent(in) *Y*, INTEGER(INTG),intent(in) *INCY*)**

Definition at line 116 of file blas.f90.

**7.14.2.5 REAL(DP) BLAS::interface::DNRM2 (INTEGER(INTG),intent(in) *N*,  
REAL(DP),dimension(\*),intent(in) *X*, INTEGER(INTG),intent(in) *INCX*)**

Definition at line 134 of file blas.f90.

**7.14.2.6 subroutine BLAS::interface::DROT (INTEGER(INTG),intent(in) *N*,  
REAL(DP),dimension(\*),intent(out) *DX*, INTEGER(INTG),intent(in) *INCX*,  
REAL(DP),dimension(\*),intent(out) *DY*, INTEGER(INTG),intent(in) *INCY*,  
REAL(DP),intent(in) *C*, REAL(DP),intent(in) *S*)**

Definition at line 153 of file blas.f90.

**7.14.2.7 subroutine BLAS::interface::DROTG (REAL(DP),intent(in) *DA*, REAL(DP),intent(in) *DB*,  
REAL(DP),intent(in) *C*, REAL(DP),intent(in) *S*)**

Definition at line 172 of file blas.f90.

**7.14.2.8 subroutine BLAS::interface::DSCAL (INTEGER(INTG),intent(in)  
*N*, REAL(DP),intent(in) *A*, REAL(DP),dimension(\*),intent(inout) *X*,  
INTEGER(INTG),intent(in) *INCX*)**

Definition at line 188 of file blas.f90.

**7.14.2.9 subroutine BLAS::interface::DTRSV (CHARACTER(LEN=1),intent(in) *UPLO*,  
CHARACTER(LEN=1),intent(in) *TRANS*, CHARACTER(LEN=1),intent(in)  
*DIAG*, INTEGER(INTG),intent(in) *N*, REAL(DP),dimension(*lda*, \*),intent(in)  
*A*, INTEGER(INTG),intent(in) *LDA*, REAL(DP),dimension(\*),intent(in) *X*,  
INTEGER(INTG),intent(in) *INCX*)**

Definition at line 237 of file blas.f90.

**7.14.2.10 subroutine BLAS::interface::SASUM (INTEGER(INTG),intent(in) *N*,  
REAL(SP),dimension(\*),intent(in) *X*, INTEGER(INTG),intent(in) *INCX*)**

Definition at line 54 of file blas.f90.

**7.14.2.11 subroutine BLAS::interface::SAXPY (INTEGER(INTG),intent(in)  
*N*, REAL(SP),intent(in) *A*, REAL(SP),dimension(\*),intent(in) *X*,  
INTEGER(INTG),intent(in) *INCX*, REAL(SP),dimension(\*),intent(out) *Y*,  
INTEGER(INTG),intent(in) *INCY*)**

Definition at line 68 of file blas.f90.

**7.14.2.12 subroutine BLAS::interface::SCOPY (INTEGER(INTG),intent(in) *N*,  
REAL(SP),dimension(\*),intent(in) *DX*, INTEGER(INTG),intent(in) *INCX*,  
REAL(SP),dimension(\*),intent(out) *DY*, INTEGER(INTG),intent(in) *INCY*)**

Definition at line 88 of file blas.f90.

**7.14.2.13 REAL(SP) BLAS::interface::SDOT (INTEGER(INTG),intent(in) *N*,  
REAL(SP),dimension(\*),intent(in) *X*, INTEGER(INTG),intent(in) *INCX*,  
REAL(SP),dimension(\*),intent(in) *Y*, INTEGER(INTG),intent(in) *INCY*)**

Definition at line 106 of file blas.f90.

**7.14.2.14 REAL(SP) BLAS::interface::SNRM2 (INTEGER(INTG),intent(in) *N*,  
REAL(SP),dimension(\*),intent(in) *X*, INTEGER(INTG),intent(in) *INCX*)**

Definition at line 126 of file blas.f90.

**7.14.2.15 subroutine BLAS::interface::SROT (INTEGER(INTG),intent(in) *N*,  
REAL(SP),dimension(\*),intent(out) *DX*, INTEGER(INTG),intent(in) *INCX*,  
REAL(SP),dimension(\*),intent(out) *DY*, INTEGER(INTG),intent(in) *INCY*,  
REAL(SP),intent(in) *C*, REAL(SP),intent(in) *S*)**

Definition at line 142 of file blas.f90.

**7.14.2.16 subroutine BLAS::interface::SROTG (REAL(SP),intent(in) *DA*, REAL(SP),intent(in)  
*DB*, REAL(SP),intent(in) *C*, REAL(SP),intent(in) *S*)**

Definition at line 164 of file blas.f90.

**7.14.2.17 subroutine BLAS::interface::SSCAL (INTEGER(INTG),intent(in)  
*N*, REAL(SP),intent(in) *A*, REAL(SP),dimension(\*),intent(inout) *X*,  
INTEGER(INTG),intent(in) *INCX*)**

Definition at line 180 of file blas.f90.

**7.14.2.18 subroutine BLAS::interface::STRSV (CHARACTER(LEN=1),intent(in) *UPLO*,  
CHARACTER(LEN=1),intent(in) *TRANS*, CHARACTER(LEN=1),intent(in)  
*DIAG*, INTEGER(INTG),intent(in) *N*, REAL(SP),dimension(*lda*, \*),intent(in)  
*A*, INTEGER(INTG),intent(in) *LDA*, REAL(SP),dimension(\*),intent(in) *X*,  
INTEGER(INTG),intent(in) *INCX*)**

Definition at line 228 of file blas.f90.

## 7.15 CMISS\_PARMETIS::interface Interface Reference

### Public Member Functions

- subroutine [ParMETIS\\_V3\\_PartMeshKway](#)

### Private Member Functions

- subroutine [ParMETIS\\_V3\\_PartK](#) (*vtxdist*, *xadj*, *adjncy*, *vwgt*, *adjwgt*, *wgtflag*, *numflag*, *ncon*, *nparts*, *tpwgts*, *ubvec*,*&options*, *edgecut*, *part*, *comm*)

#### 7.15.1 Detailed Description

Definition at line 61 of file cmiss\_parmetis.f90.

#### 7.15.2 Member Function Documentation

**7.15.2.1 subroutine CMISS\_PARMETIS::interface::ParMETIS\_V3\_PartK**  
(INTEGER(INTG),dimension(\*) *vtxdist*, INTEGER(INTG),dimension(\*) *xadj*,  
INTEGER(INTG),dimension(\*) *adjncy*, INTEGER(INTG),dimension(\*) *vwgt*,  
INTEGER(INTG),dimension(\*) *adjwgt*, INTEGER(INTG) *wgtflag*, INTEGER(INTG)  
*numflag*, INTEGER(INTG) *ncon*, INTEGER(INTG) *nparts*, REAL(SP),dimension(\*)  
*tpwgts*, REAL(SP),dimension(\*) *ubvec*, &,dimension(\*) *options*, INTEGER(INTG)  
*edgecut*, INTEGER(INTG),dimension(\*) *part*, INTEGER(INTG) *comm*) [private]

Definition at line 63 of file cmiss\_parmetis.f90.

**7.15.2.2 subroutine CMISS\_PARMETIS::interface::ParMETIS\_V3\_PartMeshKway ()**

Definition at line 86 of file cmiss\_parmetis.f90.

## 7.16 CMISS\_PETSC::interface Interface Reference

### Public Member Functions

- subroutine [SNESSetTolerances](#) (snes, abstol, rtol, stol, maxit, maxf, ierr)

### Private Member Functions

- subroutine [ISDestroy](#) (indexset, ierr)
- subroutine [ISColoringDestroy](#) (iscoloring, ierr)
- subroutine [ISLocalToGlobalMappingApply](#) (ctx, type, nin, idxin, nout, idxout, ierr)
- subroutine [ISLocalToGlobalMappingApplyIS](#) (ctx, isin, isout, ierr)
- subroutine [ISLocalToGlobalMappingCreate](#) (comm, N, globalnum, ctx, ierr)
- subroutine [ISLocalToGlobalMappingDestroy](#) (ctx, ierr)
- subroutine [KSPCreate](#) (comm, ksp, ierr)
- subroutine [KSPDestroy](#) (ksp, ierr)
- subroutine [KSPGetConvergedReason](#) (ksp, reason, ierr)
- subroutine [KSPGetIterationNumber](#) (ksp, its, ierr)
- subroutine [KSPGetPC](#) (ksp, pc, ierr)
- subroutine [KSPGetResidualNorm](#) (ksp, rnorm, ierr)
- subroutine [KSPSetFromOptions](#) (ksp, ierr)
- subroutine [KSPSetOperators](#) (ksp, Amat, Pmat, flag, ierr)
- subroutine [KSPSetTolerances](#) (ksp, rtol, atol, dtol, maxits, ierr)
- subroutine [KSPSetType](#) (ksp, method, ierr)
- subroutine [KSPSetUp](#) (ksp, ierr)
- subroutine [KSPSolve](#) (ksp, b, x, ierr)
- subroutine [MatAssemblyBegin](#) (A, assemblytype, ierr)
- subroutine [MatAssemblyEnd](#) (A, assemblytype, ierr)
- subroutine [MatCreate](#) (comm, A, ierr)
- subroutine [MatCreateMPIAIJ](#) (co m, localm, localn, globalm, globaln, diagnumbernzperrow, diag-numbernzeachrow, offdiagnumbernzperrow,&offdiagnumbernzeachrow, A, ierr)
- subroutine [MatCreateMPIDense](#) (comm, localm, localn, globalm, globaln, matrixdata, A, ierr)
- subroutine [MatCreateSeqAIJ](#) (comm, m, n, numbernzperrow, numbernzeachrow, A, ierr)
- subroutine [MatCreateSeqDense](#) (comm, m, n, matrixdata, A, ierr)
- subroutine [MatDestroy](#) (A, ierr)
- subroutine [MatFDColoringCreate](#) (A, iscoloring, fdcoloring, ierr)
- subroutine [MatFDColoringDestroy](#) (fdcoloring, ierr)
- subroutine [MatFDColoringSetFromOptions](#) (fdcoloring, ierr)
- subroutine [MatGetArray](#) (A, mat\_data, mat\_offset, ierr)
- subroutine [MatGetArrayF90](#) (A, mat\_data, ierr)
- subroutine [MatGetColoring](#) (A, coloring\_type, iscoloring, ierr)
- subroutine [MatGetOwnershipRange](#) (A, firstrow, lastrow, ierr)
- subroutine [MatGetValues](#) (A, m, idxm, n, idxn, values, ierr)
- subroutine [MatRestoreArray](#) (A, mat\_data, mat\_offset, ierr)
- subroutine [MatRestoreArrayF90](#) (A, mat\_data, ierr)
- subroutine [MatSetLocalToGlobalMapping](#) (A, ctx, ierr)
- subroutine [MatSetOption](#) (A, option, ierr)
- subroutine [MatSetSizes](#) (A, localm, localn, globalM, globalN, ierr)
- subroutine [MatSetValue](#) (A, row, col, value, insertmode, ierr)
- subroutine [MatSetValues](#) (A, m, mindices, n, nindices, values, insertmode, ierr)

- subroutine [MatSetValuesLocal](#) (A, m, mindices, n, nindices, values, insertmode, ierr)
- subroutine [MatSetValueLocal](#) (A, row, col, value, insertmode, ierr)
- subroutine [MatView](#) (A, v, ierr)
- subroutine [MatZeroEntries](#) (A, ierr)
- subroutine [PCSetType](#) (pc, method, ierr)
- subroutine [PetscFinalize](#) (ierr)
- subroutine [PetscInitialize](#) (file, ierr)
- subroutine [PetscLogPrintSummary](#) (comm, file, ierr)
- subroutine [SNESCreate](#) (comm, snes, ierr)
- subroutine [SNESDestroy](#) (snes, ierr)
- subroutine [SNESGetConvergedReason](#) (snes, reason, ierr)
- subroutine [SNESGetFunctionNorm](#) (snes, fnorm, ierr)
- subroutine [SNESGetIterationNumber](#) (snes, iter, ierr)
- subroutine [SNESLineSearchSet](#) (snes, func, lsctx, ierr)
- subroutine [SNESLineSearchSetParams](#) (snes, alpha, maxstep, steptol, ierr)
- subroutine [SNESSetFromOptions](#) (snes, ierr)
- subroutine [SNESSetFunction](#) (snes, f, ffunction, ctx, ierr)
- subroutine [SNESSetTrustRegionTolerance](#) (snes, trtol, ierr)
- subroutine [SNESSetType](#) (snes, method, ierr)
- subroutine [SNESSolve](#) (snes, b, x, ierr)
- subroutine [VecAssemblyBegin](#) (x, ierr)
- subroutine [VecAssemblyEnd](#) (x, ierr)
- subroutine [VecCreate](#) (comm, x, ierr)
- subroutine [VecCreateGhost](#) (comm, localm, globalm, nghost, ghosts, x, ierr)
- subroutine [VecCreateGhostWithArray](#) (comm, localm, globalm, nghost, ghosts, array, x, ierr)
- subroutine [VecCreateMPI](#) (comm, localm, globalm, x, ierr)
- subroutine [VecCreateMPIWithArray](#) (comm, localn, globaln, array, x, ierr)
- subroutine [VecCreateSeq](#) (comm, m, x, ierr)
- subroutine [VecCreateSeqWithArray](#) (comm, n, array, x, ierr)
- subroutine [VecDestroy](#) (x, ierr)
- subroutine [VecDuplicate](#) (old, new, ierr)
- subroutine [VecGetArray](#) (x, vec\_data, vec\_offset, ierr)
- subroutine [VecGetArrayF90](#) (x, vec\_data, ierr)
- subroutine [VecGetLocalSize](#) (x, size, ierr)
- subroutine [VecGetOwnershipRange](#) (x, low, high, ierr)
- subroutine [VecGetSize](#) (x, size, ierr)
- subroutine [VecGetValues](#) (x, n, indices, values, ierr)
- subroutine [VecGhostGetLocalForm](#) (g, l, ierr)
- subroutine [VecGhostRestoreLocalForm](#) (g, l, ierr)
- subroutine [VecGhostUpdateBegin](#) (x, insertmode, scattermode, ierr)
- subroutine [VecGhostUpdateEnd](#) (x, insertmode, scattermode, ierr)
- subroutine [VecRestoreArray](#) (x, vec\_data, vec\_offset, ierr)
- subroutine [VecRestoreArrayF90](#) (x, vec\_data, ierr)
- subroutine [VecSet](#) (x, value, ierr)
- subroutine [VecSetFromOptions](#) (x, ierr)
- subroutine [VecSetLocalToGlobalMapping](#) (v, ctx, ierr)
- subroutine [VecSetSizes](#) (x, localm, globalm, ierr)
- subroutine [VecSetValues](#) (x, n, indices, values, insertmode, ierr)
- subroutine [VecSetValuesLocal](#) (x, n, indices, values, insertmode, ierr)
- subroutine [VecView](#) (x, v, ierr)

### 7.16.1 Detailed Description

Definition at line 187 of file cmiss\_petsc.f90.

### 7.16.2 Member Function Documentation

**7.16.2.1 subroutine CMISS\_PETSC::interface::ISColoringDestroy (iscoloring, ierr)**  
[private]

Definition at line 194 of file cmiss\_petsc.f90.

**7.16.2.2 subroutine CMISS\_PETSC::interface::ISDestroy (indexset, ierr) [private]**

Definition at line 189 of file cmiss\_petsc.f90.

**7.16.2.3 subroutine CMISS\_PETSC::interface::ISLocalToGlobalMappingApply (ctx, type, nin, idxin, nout, idxout, ierr) [private]**

Definition at line 199 of file cmiss\_petsc.f90.

**7.16.2.4 subroutine CMISS\_PETSC::interface::ISLocalToGlobalMappingApplyIS (ctx, isin, isout, ierr) [private]**

Definition at line 209 of file cmiss\_petsc.f90.

**7.16.2.5 subroutine CMISS\_PETSC::interface::ISLocalToGlobalMappingCreate (comm, N, globalnum, ctx, ierr) [private]**

Definition at line 216 of file cmiss\_petsc.f90.

**7.16.2.6 subroutine CMISS\_PETSC::interface::ISLocalToGlobalMappingDestroy (ctx, ierr)**  
[private]

Definition at line 224 of file cmiss\_petsc.f90.

**7.16.2.7 subroutine CMISS\_PETSC::interface::KSPCreate (comm, ksp, ierr) [private]**

Definition at line 229 of file cmiss\_petsc.f90.

**7.16.2.8 subroutine CMISS\_PETSC::interface::KSPDestroy (ksp, ierr) [private]**

Definition at line 235 of file cmiss\_petsc.f90.

**7.16.2.9 subroutine CMISS\_PETSC::interface::KSPGetConvergedReason (ksp, reason, ierr)**  
[private]

Definition at line 240 of file cmiss\_petsc.f90.

**7.16.2.10 subroutine CMISS\_PETSC::interface::KSPGetIterationNumber (ksp, its, ierr)** [private]

Definition at line 246 of file cmiss\_petsc.f90.

**7.16.2.11 subroutine CMISS\_PETSC::interface::KSPGetPC (ksp, pc, ierr) [private]**

Definition at line 252 of file cmiss\_petsc.f90.

**7.16.2.12 subroutine CMISS\_PETSC::interface::KSPGetResidualNorm (ksp, rnorm, ierr)** [private]

Definition at line 258 of file cmiss\_petsc.f90.

**7.16.2.13 subroutine CMISS\_PETSC::interface::KSPSetFromOptions (ksp, ierr) [private]**

Definition at line 264 of file cmiss\_petsc.f90.

**7.16.2.14 subroutine CMISS\_PETSC::interface::KSPSetOperators (ksp, Amat, Pmat, flag, ierr)** [private]

Definition at line 269 of file cmiss\_petsc.f90.

**7.16.2.15 subroutine CMISS\_PETSC::interface::KSPSetTolerances (ksp, rtol, atol, dtol, maxits, ierr) [private]**

Definition at line 277 of file cmiss\_petsc.f90.

**7.16.2.16 subroutine CMISS\_PETSC::interface::KSPSetType (ksp, method, ierr) [private]**

Definition at line 286 of file cmiss\_petsc.f90.

**7.16.2.17 subroutine CMISS\_PETSC::interface::KSPSetUp (ksp, ierr) [private]**

Definition at line 292 of file cmiss\_petsc.f90.

**7.16.2.18 subroutine CMISS\_PETSC::interface::KSPSolve (ksp, b, x, ierr) [private]**

Definition at line 297 of file cmiss\_petsc.f90.

**7.16.2.19 subroutine CMISS\_PETSC::interface::MatAssemblyBegin (A, assemblytype, ierr)** [private]

Definition at line 304 of file cmiss\_petsc.f90.

**7.16.2.20 subroutine CMISS\_PETSC::interface::MatAssemblyEnd (A, assemblytype, ierr)** [private]

Definition at line 310 of file cmiss\_petsc.f90.

**7.16.2.21 subroutine CMISS\_PETSC::interface::MatCreate (comm, A, ierr)** [private]

Definition at line 316 of file cmiss\_petsc.f90.

**7.16.2.22 subroutine CMISS\_PETSC::interface::MatCreateMPIAIJ (co m, localm, localn, globalm, globaln, diagsnumbernzperrow, diagsnumbernzeachrow, offdiagsnumbernzperrow, & offdiagsnumbernzeachrow, A, ierr)** [private]

Definition at line 322 of file cmiss\_petsc.f90.

**7.16.2.23 subroutine CMISS\_PETSC::interface::MatCreateMPIDense (comm, localm, localn, globalm, globaln, matrixdata, A, ierr)** [private]

Definition at line 337 of file cmiss\_petsc.f90.

**7.16.2.24 subroutine CMISS\_PETSC::interface::MatCreateSeqAIJ (comm, m, n, numbernzperrow, numbernzeachrow, A, ierr)** [private]

Definition at line 348 of file cmiss\_petsc.f90.

**7.16.2.25 subroutine CMISS\_PETSC::interface::MatCreateSeqDense (comm, m, n, matrixdata, A, ierr)** [private]

Definition at line 358 of file cmiss\_petsc.f90.

**7.16.2.26 subroutine CMISS\_PETSC::interface::MatDestroy (A, ierr)** [private]

Definition at line 367 of file cmiss\_petsc.f90.

**7.16.2.27 subroutine CMISS\_PETSC::interface::MatFDColoringCreate (A, iscoloring, fdcoloring, ierr)** [private]

Definition at line 372 of file cmiss\_petsc.f90.

**7.16.2.28 subroutine CMISS\_PETSC::interface::MatFDColoringDestroy (fdcoloring, ierr)** [private]

Definition at line 379 of file cmiss\_petsc.f90.

**7.16.2.29 subroutine CMISS\_PETSC::interface::MatFDColoringSetFromOptions (fdcoloring, ierr)** [private]

Definition at line 384 of file cmiss\_petsc.f90.

**7.16.2.30 subroutine CMISS\_PETSC::interface::MatGetArray (A, mat\_data, mat\_offset, ierr)**  
[private]

Definition at line 389 of file cmiss\_petsc.f90.

**7.16.2.31 subroutine CMISS\_PETSC::interface::MatGetArrayF90 (A, mat\_data, ierr)**  
[private]

Definition at line 396 of file cmiss\_petsc.f90.

**7.16.2.32 subroutine CMISS\_PETSC::interface::MatGetColoring (A, coloring\_type, iscoloring, ierr)**  
[private]

Definition at line 402 of file cmiss\_petsc.f90.

**7.16.2.33 subroutine CMISS\_PETSC::interface::MatGetOwnershipRange (A, firstrow, lastrow, ierr)**  
[private]

Definition at line 409 of file cmiss\_petsc.f90.

**7.16.2.34 subroutine CMISS\_PETSC::interface::MatGetValues (A, m, idxm, n, idxn, values, ierr)**  
[private]

Definition at line 416 of file cmiss\_petsc.f90.

**7.16.2.35 subroutine CMISS\_PETSC::interface::MatRestoreArray (A, mat\_data, mat\_offset, ierr)**  
[private]

Definition at line 426 of file cmiss\_petsc.f90.

**7.16.2.36 subroutine CMISS\_PETSC::interface::MatRestoreArrayF90 (A, mat\_data, ierr)**  
[private]

Definition at line 433 of file cmiss\_petsc.f90.

**7.16.2.37 subroutine CMISS\_PETSC::interface::MatSetLocalToGlobalMapping (A, ctx, ierr)**  
[private]

Definition at line 439 of file cmiss\_petsc.f90.

**7.16.2.38 subroutine CMISS\_PETSC::interface::MatSetOption (A, option, ierr)** [private]

Definition at line 445 of file cmiss\_petsc.f90.

**7.16.2.39 subroutine CMISS\_PETSC::interface::MatSetSizes (A, localm, localn, globalM, globalN, ierr)**  
[private]

Definition at line 451 of file cmiss\_petsc.f90.

**7.16.2.40 subroutine CMISS\_PETSC::interface::MatSetValue (A, row, col, value, insertmode, ierr) [private]**

Definition at line 460 of file cmiss\_petsc.f90.

**7.16.2.41 subroutine CMISS\_PETSC::interface::MatSetValueLocal (A, row, col, value, insertmode, ierr) [private]**

Definition at line 491 of file cmiss\_petsc.f90.

**7.16.2.42 subroutine CMISS\_PETSC::interface::MatSetValues (A, m, mindices, n, nindices, values, insertmode, ierr) [private]**

Definition at line 469 of file cmiss\_petsc.f90.

**7.16.2.43 subroutine CMISS\_PETSC::interface::MatSetValuesLocal (A, m, mindices, n, nindices, values, insertmode, ierr) [private]**

Definition at line 480 of file cmiss\_petsc.f90.

**7.16.2.44 subroutine CMISS\_PETSC::interface::MatView (A, v, ierr) [private]**

Definition at line 500 of file cmiss\_petsc.f90.

**7.16.2.45 subroutine CMISS\_PETSC::interface::MatZeroEntries (A, ierr) [private]**

Definition at line 506 of file cmiss\_petsc.f90.

**7.16.2.46 subroutine CMISS\_PETSC::interface::PCSetType (pc, method, ierr) [private]**

Definition at line 511 of file cmiss\_petsc.f90.

**7.16.2.47 subroutine CMISS\_PETSC::interface::PetscFinalize (ierr) [private]**

Definition at line 517 of file cmiss\_petsc.f90.

**7.16.2.48 subroutine CMISS\_PETSC::interface::PetscInitialize (file, ierr) [private]**

Definition at line 521 of file cmiss\_petsc.f90.

**7.16.2.49 subroutine CMISS\_PETSC::interface::PetscLogPrintSummary (comm, file, ierr) [private]**

Definition at line 526 of file cmiss\_petsc.f90.

**7.16.2.50 subroutine CMISS\_PETSC::interface::SNESCreate (comm, snes, ierr) [private]**

Definition at line 532 of file cmiss\_petsc.f90.

**7.16.2.51 subroutine CMISS\_PETSC::interface::SNESDestroy (snes, ierr) [private]**

Definition at line 538 of file cmiss\_petsc.f90.

**7.16.2.52 subroutine CMISS\_PETSC::interface::SNESGetConvergedReason (snes, reason, ierr) [private]**

Definition at line 543 of file cmiss\_petsc.f90.

**7.16.2.53 subroutine CMISS\_PETSC::interface::SNESGetFunctionNorm (snes, fnorm, ierr) [private]**

Definition at line 549 of file cmiss\_petsc.f90.

**7.16.2.54 subroutine CMISS\_PETSC::interface::SNESGetIterationNumber (snes, iter, ierr) [private]**

Definition at line 555 of file cmiss\_petsc.f90.

**7.16.2.55 subroutine CMISS\_PETSC::interface::SNESLineSearchSet (snes, func, lsctx, ierr) [private]**

Definition at line 561 of file cmiss\_petsc.f90.

**7.16.2.56 subroutine CMISS\_PETSC::interface::SNESLineSearchSetParams (snes, alpha, maxstep, steptol, ierr) [private]**

Definition at line 568 of file cmiss\_petsc.f90.

**7.16.2.57 subroutine CMISS\_PETSC::interface::SNESSetFromOptions (snes, ierr) [private]**

Definition at line 576 of file cmiss\_petsc.f90.

**7.16.2.58 subroutine CMISS\_PETSC::interface::SNESSetFunction (snes, f, ffunction, TYPE(SOLUTION\_TYPE),pointer ctx, ierr) [private]**

Definition at line 581 of file cmiss\_petsc.f90.

**7.16.2.59 subroutine CMISS\_PETSC::interface::SNESSetTolerances (snes, abstol, rtol, stol, maxit, maxf, ierr)**

Definition at line 601 of file cmiss\_petsc.f90.

**7.16.2.60 subroutine CMISS\_PETSC::interface::SNESSetTrustRegionTolerance (snes, trtol, ierr) [private]**

Definition at line 611 of file cmiss\_petsc.f90.

**7.16.2.61 subroutine CMISS\_PETSC::interface::SNESSetType (snes, method, ierr) [private]**

Definition at line 617 of file cmiss\_petsc.f90.

**7.16.2.62 subroutine CMISS\_PETSC::interface::SNESSolve (snes, b, x, ierr) [private]**

Definition at line 623 of file cmiss\_petsc.f90.

**7.16.2.63 subroutine CMISS\_PETSC::interface::VecAssemblyBegin (x, ierr) [private]**

Definition at line 630 of file cmiss\_petsc.f90.

**7.16.2.64 subroutine CMISS\_PETSC::interface::VecAssemblyEnd (x, ierr) [private]**

Definition at line 635 of file cmiss\_petsc.f90.

**7.16.2.65 subroutine CMISS\_PETSC::interface::VecCreate (comm, x, ierr) [private]**

Definition at line 640 of file cmiss\_petsc.f90.

**7.16.2.66 subroutine CMISS\_PETSC::interface::VecCreateGhost (comm, localm, globalm, nghost, ghosts, x, ierr) [private]**

Definition at line 646 of file cmiss\_petsc.f90.

**7.16.2.67 subroutine CMISS\_PETSC::interface::VecCreateGhostWithArray (comm, localm, globalm, nghost, ghosts, array, x, ierr) [private]**

Definition at line 656 of file cmiss\_petsc.f90.

**7.16.2.68 subroutine CMISS\_PETSC::interface::VecCreateMPI (comm, localm, globalm, x, ierr) [private]**

Definition at line 667 of file cmiss\_petsc.f90.

**7.16.2.69 subroutine CMISS\_PETSC::interface::VecCreateMPIWithArray (comm, localn, globaln, array, x, ierr) [private]**

Definition at line 675 of file cmiss\_petsc.f90.

**7.16.2.70 subroutine CMISS\_PETSC::interface::VecCreateSeq (comm, m, x, ierr) [private]**

Definition at line 684 of file cmiss\_petsc.f90.

**7.16.2.71 subroutine CMISS\_PETSC::interface::VecCreateSeqWithArray (comm, n, array, x, ierr) [private]**

Definition at line 691 of file cmiss\_petsc.f90.

**7.16.2.72 subroutine CMISS\_PETSC::interface::VecDestroy (x, ierr) [private]**

Definition at line 699 of file cmiss\_petsc.f90.

**7.16.2.73 subroutine CMISS\_PETSC::interface::VecDuplicate (old, new, ierr) [private]**

Definition at line 704 of file cmiss\_petsc.f90.

**7.16.2.74 subroutine CMISS\_PETSC::interface::VecGetArray (x, vec\_data, vec\_offset, ierr) [private]**

Definition at line 709 of file cmiss\_petsc.f90.

**7.16.2.75 subroutine CMISS\_PETSC::interface::VecGetArrayF90 (x, vec\_data, ierr) [private]**

Definition at line 716 of file cmiss\_petsc.f90.

**7.16.2.76 subroutine CMISS\_PETSC::interface::VecGetLocalSize (x, size, ierr) [private]**

Definition at line 722 of file cmiss\_petsc.f90.

**7.16.2.77 subroutine CMISS\_PETSC::interface::VecGetOwnershipRange (x, low, high, ierr) [private]**

Definition at line 728 of file cmiss\_petsc.f90.

**7.16.2.78 subroutine CMISS\_PETSC::interface::VecGetSize (x, size, ierr) [private]**

Definition at line 735 of file cmiss\_petsc.f90.

**7.16.2.79 subroutine CMISS\_PETSC::interface::VecGetValues (x, n, indices, values, ierr) [private]**

Definition at line 741 of file cmiss\_petsc.f90.

**7.16.2.80 subroutine CMISS\_PETSC::interface::VecGhostGetLocalForm (g, l, ierr)**  
[private]

Definition at line 749 of file cmiss\_petsc.f90.

**7.16.2.81 subroutine CMISS\_PETSC::interface::VecGhostRestoreLocalForm (g, l, ierr)**  
[private]

Definition at line 755 of file cmiss\_petsc.f90.

**7.16.2.82 subroutine CMISS\_PETSC::interface::VecGhostUpdateBegin (x, insertmode, scattermode, ierr) [private]**

Definition at line 761 of file cmiss\_petsc.f90.

**7.16.2.83 subroutine CMISS\_PETSC::interface::VecGhostUpdateEnd (x, insertmode, scattermode, ierr) [private]**

Definition at line 768 of file cmiss\_petsc.f90.

**7.16.2.84 subroutine CMISS\_PETSC::interface::VecRestoreArray (x, vec\_data, vec\_offset, ierr)**  
[private]

Definition at line 775 of file cmiss\_petsc.f90.

**7.16.2.85 subroutine CMISS\_PETSC::interface::VecRestoreArrayF90 (x, vec\_data, ierr)**  
[private]

Definition at line 782 of file cmiss\_petsc.f90.

**7.16.2.86 subroutine CMISS\_PETSC::interface::VecSet (x, value, ierr) [private]**

Definition at line 788 of file cmiss\_petsc.f90.

**7.16.2.87 subroutine CMISS\_PETSC::interface::VecSetFromOptions (x, ierr) [private]**

Definition at line 794 of file cmiss\_petsc.f90.

**7.16.2.88 subroutine CMISS\_PETSC::interface::VecSetLocalToGlobalMapping (v, ctx, ierr)**  
[private]

Definition at line 799 of file cmiss\_petsc.f90.

**7.16.2.89 subroutine CMISS\_PETSC::interface::VecSetSizes (x, localm, globalm, ierr)**  
[private]

Definition at line 805 of file cmiss\_petsc.f90.

**7.16.2.90 subroutine CMISS\_PETSC::interface::VecSetValues (x, n, indices, values, insertmode, ierr) [private]**

Definition at line 811 of file cmiss\_petsc.f90.

**7.16.2.91 subroutine CMISS\_PETSC::interface::VecSetValuesLocal (x, n, indices, values, insertmode, ierr) [private]**

Definition at line 820 of file cmiss\_petsc.f90.

**7.16.2.92 subroutine CMISS\_PETSC::interface::VecView (x, v, ierr) [private]**

Definition at line 829 of file cmiss\_petsc.f90.

## 7.17 CMISS\_PETSC::PETSC\_SNESSETJACOBIAN Interface Reference

### Private Member Functions

- subroutine [PETSC\\_SNESSETJACOBIAN\\_SOLVER](#) (SNES\_, A, B, JFUNCTION, CTX, ERR, ERROR,\*)
- subroutine [PETSC\\_SNESSETJACOBIAN\\_MATFDCOLORING](#) (SNES\_, A, B, JFUNCTION, CTX, ERR, ERROR,\*)

#### 7.17.1 Detailed Description

Definition at line 837 of file cmiss\_petsc.f90.

#### 7.17.2 Member Function Documentation

**7.17.2.1 subroutine CMISS\_PETSC::PETSC\_SNESSETJACOBIAN::PETSC\_SNESSETJACOBIAN\_MATFDCOLORING** (TYPE(PETSC\_SNES\_TYPE),intent(inout) SNES\_, TYPE(PETSC\_MAT\_TYPE),intent(inout) A, TYPE(PETSC\_MAT\_TYPE),intent(inout) B, JFUNCTION, TYPE(PETSC\_MATFDCOLORING\_TYPE) CTX, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]

##### Parameters:

**SNES\_** The SNES to set the function for  
**A** The Jacobian matrix  
**B** The Jacobian preconditioning matrix  
**CTX** The MatFDColoring data to pass to the function  
**ERR** The error code  
**ERROR** The error string

Definition at line 3173 of file cmiss\_petsc.f90.

**7.17.2.2 subroutine CMISS\_PETSC::PETSC\_SNESSETJACOBIAN::PETSC\_SNESSETJACOBIAN\_SOLVER** (TYPE(PETSC\_SNES\_TYPE),intent(inout) SNES\_, TYPE(PETSC\_MAT\_TYPE),intent(inout) A, TYPE(PETSC\_MAT\_TYPE),intent(inout) B, JFUNCTION, TYPE(SOLVER\_TYPE),pointer CTX, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]

##### Parameters:

**SNES\_** The SNES to set the function for  
**A** The Jacobian matrix  
**B** The Jacobian preconditioning matrix  
**CTX** The solver data to pass to the function  
**ERR** The error code  
**ERROR** The error string

Definition at line 3207 of file cmiss\_petsc.f90.

## 7.18 CMISS\_PETSC\_TYPES::PETSC\_IS\_TYPE Struct Reference

### 7.18.1 Detailed Description

Definition at line 65 of file cmiss\_petsc\_types.f90.

## 7.19 CMISS\_PETSC\_TYPES::PETSC\_ISCOLORING\_TYPE Struct Reference

### 7.19.1 Detailed Description

Definition at line 73 of file cmiss\_petsc\_types.f90.

## 7.20 CMISS\_PETSC\_TYPES::PETSC\_ISLOCALTOGLOBALMAPPING\_- TYPE Struct Reference

### 7.20.1 Detailed Description

Definition at line 69 of file cmiss\_petsc\_types.f90.

## 7.21 CMISS\_PETSC\_TYPES::PETSC\_KSP\_TYPE Struct Reference

### 7.21.1 Detailed Description

Definition at line 77 of file cmiss\_petsc\_types.f90.

## 7.22 CMISS\_PETSC\_TYPES::PETSC\_MAT\_TYPE Struct Reference

### 7.22.1 Detailed Description

Definition at line 81 of file cmiss\_petsc\_types.f90.

## 7.23 CMISS\_PETSC\_TYPES::PETSC\_MATFDCOLORING\_- TYPE Struct Reference

### 7.23.1 Detailed Description

Definition at line 87 of file cmiss\_petsc\_types.f90.

## 7.24 CMISS\_PETSC\_TYPES::PETSC\_PC\_TYPE Struct Reference

### 7.24.1 Detailed Description

Definition at line 91 of file cmiss\_petsc\_types.f90.

## 7.25 CMISS\_PETSC\_TYPES::PETSC\_SNES\_TYPE Struct Reference

### 7.25.1 Detailed Description

Definition at line 95 of file cmiss\_petsc\_types.f90.

## 7.26 CMISS\_PETSC\_TYPES::PETSC\_VEC\_TYPE Struct Reference

### 7.26.1 Detailed Description

Definition at line 99 of file cmiss\_petsc\_types.f90.

## 7.27 COMP\_ENVIRONMENT::CACHE\_TYPE Struct Reference

Contains information on a cache hierarchy.

### Private Attributes

- INTEGER(INTG) **NUMBER\_LEVELS**  
*The number of levels in the cache hierarchy.*
- INTEGER(INTG), allocatable **SIZE**  
*SIZE(level\_idx). The size of the level\_idx'th cache level.*

#### 7.27.1 Detailed Description

Contains information on a cache hierarchy.

Definition at line 64 of file computational\_environment.f90.

#### 7.27.2 Member Data Documentation

##### 7.27.2.1 INTEGER(INTG) COMP\_ENVIRONMENT::CACHE\_TYPE::NUMBER\_LEVELS [private]

The number of levels in the cache hierarchy.

Definition at line 65 of file computational\_environment.f90.

##### 7.27.2.2 INTEGER(INTG),allocatable COMP\_ENVIRONMENT::CACHE\_TYPE::SIZE [private]

SIZE(level\_idx). The size of the level\_idx'th cache level.

Definition at line 66 of file computational\_environment.f90.

## 7.28 COMP\_ENVIRONMENT::COMPUTATIONAL\_- ENVIRONMENT\_TYPE Struct Reference

Contains information on the computational environment the program is running in.

Collaboration diagram for COMP\_ENVIRONMENT::COMPUTATIONAL\_ENVIRONMENT\_TYPE:

### Private Attributes

- INTEGER(INTG) [MPI\\_COMM](#)  
*The MPI communicator for cmiss.*
- INTEGER(INTG) [NUMBER\\_COMPUTATIONAL\\_NODES](#)  
*The number of computational nodes.*
- INTEGER(INTG) [MY\\_COMPUTATIONAL\\_NODE\\_NUMBER](#)  
*The index of the running process.*
- TYPE([COMPUTATIONAL\\_NODE\\_TYPE](#)), allocatable [COMPUTATIONAL\\_NODES](#)  
*COMPUTATIONAL\_NODES(node\_idx). Contains information on the node\_idx'th computational node.*

### 7.28.1 Detailed Description

Contains information on the computational environment the program is running in.

Definition at line 88 of file computational\_environment.f90.

### 7.28.2 Member Data Documentation

#### 7.28.2.1 TYPE(COMPUTATIONAL\_NODE\_TYPE),allocatable COMP\_- ENVIRONMENT::COMPUTATIONAL\_ENVIRONMENT\_- TYPE::COMPUTATIONAL\_NODES [private]

COMPUTATIONAL\_NODES(node\_idx). Contains information on the node\_idx'th computational node.

Definition at line 92 of file computational\_environment.f90.

#### 7.28.2.2 INTEGER(INTG) COMP\_ENVIRONMENT::COMPUTATIONAL\_ENVIRONMENT\_- TYPE::MPI\_COMM [private]

The MPI communicator for cmiss.

Definition at line 89 of file computational\_environment.f90.

#### 7.28.2.3 INTEGER(INTG) COMP\_ENVIRONMENT::COMPUTATIONAL\_- ENVIRONMENT\_TYPE::MY\_COMPUTATIONAL\_NODE\_NUMBER [private]

The index of the running process.

Definition at line 91 of file computational\_environment.f90.

**7.28.2.4 INTEGER(INTG) COMP\_ENVIRONMENT::COMPUTATIONAL\_ENVIRONMENT\_TYPE::NUMBER\_COMPUTATIONAL\_NODES  
[private]**

The number of computational nodes.

Definition at line 90 of file computational\_environment.f90.

## 7.29 COMP\_ENVIRONMENT::COMPUTATIONAL\_NODE\_- TYPE Struct Reference

Contains information on a computational node containing a number of processors.

### Private Attributes

- INTEGER(INTG) [NUMBER\\_PROCESSORS](#)  
*The number of processors for this computational node.*
- INTEGER(INTG) [RANK](#)  
*The MPI rank of this computational node.*
- INTEGER(INTG) [NODE\\_NAME\\_LENGTH](#)  
*The length of the name of the computational node.*
- CHARACTER(LEN=MPI\_MAX\_PROCESSOR\_NAME) [NODE\\_NAME](#)  
*The name of the computational node.*

### 7.29.1 Detailed Description

Contains information on a computational node containing a number of processors.

Definition at line 70 of file computational\_environment.f90.

### 7.29.2 Member Data Documentation

#### 7.29.2.1 CHARACTER(LEN=MPI\_MAX\_PROCESSOR\_NAME) COMP\_- ENVIRONMENT::COMPUTATIONAL\_NODE\_TYPE::NODE\_NAME [private]

The name of the computational node.

Definition at line 75 of file computational\_environment.f90.

#### 7.29.2.2 INTEGER(INTG) COMP\_ENVIRONMENT::COMPUTATIONAL\_NODE\_- TYPE::NODE\_NAME\_LENGTH [private]

The length of the name of the computational node.

Definition at line 74 of file computational\_environment.f90.

#### 7.29.2.3 INTEGER(INTG) COMP\_ENVIRONMENT::COMPUTATIONAL\_NODE\_- TYPE::NUMBER\_PROCESSORS [private]

The number of processors for this computational node.

Definition at line 71 of file computational\_environment.f90.

**7.29.2.4 INTEGER(INTG) COMP\_ENVIRONMENT::COMPUTATIONAL\_NODE\_-  
TYPE::RANK [private]**

The MPI rank of this computational node.

Definition at line 72 of file computational\_environment.f90.

## 7.30 COMP\_ENVIRONMENT::MPI\_COMPUTATIONAL\_NODE\_TYPE Struct Reference

Contains information on the MPI type to transfer information about a computational node.

### Private Attributes

- INTEGER(INTG) [MPI\\_TYPE](#)  
*The MPI data type.*
- INTEGER(INTG) [NUM\\_BLOCKS](#)  
*The number of blocks in the MPI data type. This will be equal to 4.*
- INTEGER(INTG) [BLOCK\\_LENGTHS](#)  
*The length of each block.*
- INTEGER(INTG) [TYPES](#)  
*The data types of each block.*
- INTEGER(MPI\_ADDRESS\_KIND) [DISPLACEMENTS](#)  
*The address displacements to each block.*

### 7.30.1 Detailed Description

Contains information on the MPI type to transfer information about a computational node.

Definition at line 79 of file computational\_environment.f90.

### 7.30.2 Member Data Documentation

#### 7.30.2.1 INTEGER(INTG) COMP\_ENVIRONMENT::MPI\_COMPUTATIONAL\_NODE\_TYPE::BLOCK\_LENGTHS [private]

The length of each block.

Definition at line 82 of file computational\_environment.f90.

#### 7.30.2.2 INTEGER(MPI\_ADDRESS\_KIND) COMP\_ENVIRONMENT::MPI\_COMPUTATIONAL\_NODE\_TYPE::DISPLACEMENTS [private]

The address displacements to each block.

Definition at line 84 of file computational\_environment.f90.

#### 7.30.2.3 INTEGER(INTG) COMP\_ENVIRONMENT::MPI\_COMPUTATIONAL\_NODE\_TYPE::MPI\_TYPE [private]

The MPI data type.

Definition at line 80 of file computational\_environment.f90.

**7.30.2.4 INTEGER(INTG) COMP\_ENVIRONMENT::MPI\_COMPUTATIONAL\_NODE\_-  
TYPE::NUM\_BLOCKS [private]**

The number of blocks in the MPI data type. This will be equal to 4.

Definition at line 81 of file computational\_environment.f90.

**7.30.2.5 INTEGER(INTG) COMP\_ENVIRONMENT::MPI\_COMPUTATIONAL\_NODE\_-  
TYPE::TYPES [private]**

The data types of each block.

Definition at line 83 of file computational\_environment.f90.

## 7.31 COORDINATE\_ROUTINES::COORDINATE\_CONVERT\_-- FROM\_RC Interface Reference

### Private Member Functions

- REAL(DP) [COORDINATE\\_CONVERT\\_FROM\\_RC\\_DP](#) (COORDINATE\_SYSTEM, Z, ERR, ERROR)
- REAL(SP) [COORDINATE\\_CONVERT\\_FROM\\_RC\\_SP](#) (COORDINATE\_SYSTEM, Z, ERR, ERROR)

#### 7.31.1 Detailed Description

Definition at line 119 of file coordinate\_routines.f90.

#### 7.31.2 Member Function Documentation

##### 7.31.2.1 REAL(DP) COORDINATE\_ROUTINES::COORDINATE\_-- CONVERT\_FROM\_RC::COORDINATE\_CONVERT\_FROM\_RC\_DP (TYPE(COORDINATE\_SYSTEM\_TYPE),intent(in) COORDINATE\_SYSTEM, REAL(DP),dimension(:,),intent(in) Z, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]

Definition at line 227 of file coordinate\_routines.f90.

##### 7.31.2.2 REAL(SP) COORDINATE\_ROUTINES::COORDINATE\_-- CONVERT\_FROM\_RC::COORDINATE\_CONVERT\_FROM\_RC\_SP (TYPE(COORDINATE\_SYSTEM\_TYPE),intent(in) COORDINATE\_SYSTEM, REAL(SP),dimension(:,),intent(in) Z, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]

Definition at line 352 of file coordinate\_routines.f90.

## **7.32 COORDINATE\_ROUTINES::COORDINATE\_CONVERT\_TO\_RC Interface Reference**

### **Private Member Functions**

- REAL(DP) [COORDINATE\\_CONVERT\\_TO\\_RC\\_DP](#) (COORDINATE\_SYSTEM, X, ERR, ERROR)
- REAL(SP) [COORDINATE\\_CONVERT\\_TO\\_RC\\_SP](#) (COORDINATE\_SYSTEM, X, ERR, ERROR)

#### **7.32.1 Detailed Description**

Definition at line 124 of file coordinate\_routines.f90.

#### **7.32.2 Member Function Documentation**

**7.32.2.1 REAL(DP) COORDINATE\_ROUTINES::COORDINATE\_CONVERT\_TO\_RC::COORDINATE\_CONVERT\_TO\_RC\_DP**  
(**TYPE(COORDINATE\_SYSTEM\_TYPE),intent(in) COORDINATE\_SYSTEM,**  
**REAL(DP),dimension(:),intent(in) X, INTEGER(INTG),intent(out) ERR,**  
**TYPE(VARYING\_STRING),intent(out) ERROR** [private])

Definition at line 488 of file coordinate\_routines.f90.

**7.32.2.2 REAL(SP) COORDINATE\_ROUTINES::COORDINATE\_CONVERT\_TO\_RC::COORDINATE\_CONVERT\_TO\_RC\_SP**  
(**TYPE(COORDINATE\_SYSTEM\_TYPE),intent(in) COORDINATE\_SYSTEM,**  
**REAL(SP),dimension(:),intent(in) X, INTEGER(INTG),intent(out) ERR,**  
**TYPE(VARYING\_STRING),intent(out) ERROR** [private])

Definition at line 571 of file coordinate\_routines.f90.

## 7.33 COORDINATE\_ROUTINES::COORDINATE\_DELTA\_- CALCULATE Interface Reference

### Private Member Functions

- REAL(DP) [COORDINATE\\_DELTA\\_CALCULATE\\_DP](#) (COORDINATE\_SYSTEM, X, Y, ERR, ERROR)

#### 7.33.1 Detailed Description

Definition at line 129 of file coordinate\_routines.f90.

#### 7.33.2 Member Function Documentation

- 7.33.2.1 REAL(DP) [COORDINATE\\_ROUTINES::COORDINATE\\_DELTA\\_-  
CALCULATE::COORDINATE\\_DELTA\\_CALCULATE\\_DP](#)  
(TYPE(COORDINATE\_SYSTEM\_TYPE),intent(in) *COORDINATE\_SYSTEM*,  
REAL(DP),dimension(:),intent(in) *X*, REAL(DP),dimension(:),intent(in) *Y*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*)  
[private]

Definition at line 664 of file coordinate\_routines.f90.

## 7.34 COORDINATE\_ROUTINES::COORDINATE\_- DERIVATIVE\_CONVERT\_TO\_RC Interface Reference

### Private Member Functions

- COORDINATE\_DERIVATIVE\_CONVERT\_TO\_RC\_DP
- COORDINATE\_DERIVATIVE\_CONVERT\_TO\_RC\_SP

#### 7.34.1 Detailed Description

Definition at line 182 of file coordinate\_routines.f90.

#### 7.34.2 Member Function Documentation

- 7.34.2.1 COORDINATE\_ROUTINES::COORDINATE\_DERIVATIVE\_CONVERT\_-  
TO\_RC::COORDINATE\_DERIVATIVE\_CONVERT\_TO\_RC\_DP 0  
[private]
- 7.34.2.2 COORDINATE\_ROUTINES::COORDINATE\_DERIVATIVE\_CONVERT\_-  
TO\_RC::COORDINATE\_DERIVATIVE\_CONVERT\_TO\_RC\_SP 0  
[private]

## 7.35 COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_- DESTROY Interface Reference

### Private Member Functions

- subroutine `COORDINATE_SYSTEM_DESTROY_NUMBER` (`USER_NUMBER`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_DESTROY_PTR` (`COORDINATE_SYSTEM`, `ERR`, `ERROR,*`)

#### 7.35.1 Detailed Description

Definition at line 187 of file coordinate\_routines.f90.

#### 7.35.2 Member Function Documentation

7.35.2.1 subroutine `COORDINATE_ROUTINES::COORDINATE_SYSTEM_-  
DESTROY::COORDINATE_SYSTEM_DESTROY_NUMBER`  
(`INTEGER(INTG) USER_NUMBER`, `INTEGER(INTG),intent(out) ERR`,  
`TYPE(VARYING_STRING),intent(out) ERROR, *`) [private]

Definition at line 1818 of file coordinate\_routines.f90.

7.35.2.2 subroutine `COORDINATE_ROUTINES::COORDINATE_-  
SYSTEM_DESTROY::COORDINATE_SYSTEM_DESTROY_PTR`  
(`TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_SYSTEM`,  
`INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, *`)  
[private]

Definition at line 1874 of file coordinate\_routines.f90.

## 7.36 COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_- DIMENSION\_SET Interface Reference

### Private Member Functions

- subroutine `COORDINATE_SYSTEM_DIMENSION_SET_NUMBER` (`USER_NUMBER`, `DIMENSION`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_DIMENSION_SET_PTR` (`COORDINATE_SYSTEM`, `DIMENSION`, `ERR`, `ERROR,*`)

#### 7.36.1 Detailed Description

Definition at line 134 of file coordinate\_routines.f90.

#### 7.36.2 Member Function Documentation

##### 7.36.2.1 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_- DIMENSION\_SET::COORDINATE\_SYSTEM\_DIMENSION\_SET\_NUMBER (`INTEGER(INTG),intent(in)` `USER_NUMBER`, `INTEGER(INTG),intent(in)` `DIMENSION`, `INTEGER(INTG),intent(out)` `ERR`, `TYPE(VARYING_-` `STRING),intent(out)` `ERROR`, `*`) [private]

Definition at line 1180 of file coordinate\_routines.f90.

##### 7.36.2.2 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_- DIMENSION\_SET::COORDINATE\_SYSTEM\_DIMENSION\_SET\_PTR (`TYPE(COORDINATE_SYSTEM_TYPE),pointer` `COORDINATE_SYSTEM`, `INTEGER(INTG),intent(in)` `DIMENSION`, `INTEGER(INTG),intent(out)` `ERR`, `TYPE(VARYING_STRING),intent(out)` `ERROR`, `*`) [private]

Definition at line 1211 of file coordinate\_routines.f90.

## 7.37 COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_-FOCUS\_SET Interface Reference

### Private Member Functions

- subroutine `COORDINATE_SYSTEM_FOCUS_SET_NUMBER` (`USER_NUMBER`, `FOCUS`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_FOCUS_SET_PTR` (`COORDINATE_SYSTEM`, `FOCUS`, `ERR`, `ERROR,*`)

#### 7.37.1 Detailed Description

Definition at line 139 of file coordinate\_routines.f90.

#### 7.37.2 Member Function Documentation

7.37.2.1 subroutine `COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET::COORDINATE_SYSTEM_FOCUS_SET_NUMBER` (`INTEGER(INTG),intent(in) USER_NUMBER`, `REAL(DP),intent(in) FOCUS`, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, *`) [private]

Definition at line 1281 of file coordinate\_routines.f90.

7.37.2.2 subroutine `COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET::COORDINATE_SYSTEM_FOCUS_SET_PTR` (`TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_SYSTEM`, `REAL(DP),intent(in) FOCUS`, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, *`) [private]

Definition at line 1312 of file coordinate\_routines.f90.

## 7.38 COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_ORIENTATION\_SET Interface Reference

### Private Member Functions

- subroutine `COORDINATE_SYSTEM_ORIENTATION_SET_NUMBER` (`USER_NUMBER`, `ORIENTATION`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_ORIENTATION_SET_PTR` (`COORDINATE_SYSTEM`, `ORIENTATION`, `ERR`, `ERROR,*`)

#### 7.38.1 Detailed Description

Definition at line 159 of file coordinate\_routines.f90.

#### 7.38.2 Member Function Documentation

7.38.2.1 subroutine `COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIENTATION_SET::COORDINATE_SYSTEM_ORIENTATION_SET_NUMBER` (`INTEGER(INTG),intent(in) USER_NUMBER`, `REAL(DP),dimension(:, :, intent(in)) ORIENTATION`, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, *`) [private]

Definition at line 1624 of file coordinate\_routines.f90.

7.38.2.2 subroutine `COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIENTATION_SET::COORDINATE_SYSTEM_ORIENTATION_SET_PTR` (`TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_SYSTEM`, `REAL(DP),dimension(:, :, intent(in)) ORIENTATION`, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, *`) [private]

Definition at line 1655 of file coordinate\_routines.f90.

## 7.39 COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_- ORIGIN\_SET Interface Reference

### Private Member Functions

- subroutine `COORDINATE_SYSTEM_ORIGIN_SET_NUMBER` (`USER_NUMBER`, `ORIGIN`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_ORIGIN_SET_PTR` (`COORDINATE_SYSTEM`, `ORIGIN`, `ERR`, `ERROR,*`)

#### 7.39.1 Detailed Description

Definition at line 154 of file coordinate\_routines.f90.

#### 7.39.2 Member Function Documentation

7.39.2.1 subroutine `COORDINATE_ROUTINES::COORDINATE_SYSTEM_-  
ORIGIN_SET::COORDINATE_SYSTEM_ORIGIN_SET_NUMBER`  
(`INTEGER(INTG),intent(in) USER_NUMBER`, `REAL(DP),dimension(:),intent(in)`  
`ORIGIN`, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out)`  
`ERROR, * ) [private]`

Definition at line 1552 of file coordinate\_routines.f90.

7.39.2.2 subroutine `COORDINATE_ROUTINES::COORDINATE_SYSTEM_-  
ORIGIN_SET::COORDINATE_SYSTEM_ORIGIN_SET_PTR`  
(`TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_SYSTEM,`  
`REAL(DP),dimension(:),intent(in) ORIGIN`, `INTEGER(INTG),intent(out) ERR`,  
`TYPE(VARYING_STRING),intent(out) ERROR, * ) [private]`

Definition at line 1583 of file coordinate\_routines.f90.

## **7.40 COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_PTR\_TYPE Struct Reference**

Collaboration diagram for COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_PTR\_TYPE:

### **Private Attributes**

- TYPE(COORDINATE\_SYSTEM\_TYPE), pointer PTR

#### **7.40.1 Detailed Description**

Definition at line 95 of file coordinate\_routines.f90.

#### **7.40.2 Member Data Documentation**

##### **7.40.2.1 TYPE(COORDINATE\_SYSTEM\_TYPE),pointer COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_PTR\_TYPE::PTR [private]**

Definition at line 96 of file coordinate\_routines.f90.

## 7.41 COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_- RADIAL\_INTERPOLATION\_TYPE\_SET Interface Reference

### Private Member Functions

- subroutine `COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_NUMBER`  
(USER\_NUMBER, RADIAL\_INTERPOLATION\_TYPE, ERR, ERROR,\*)
- subroutine `COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR`  
(COORDINATE\_SYSTEM, RADIAL\_INTERPOLATION\_TYPE, ERR, ERROR,\*)

#### 7.41.1 Detailed Description

Definition at line 144 of file coordinate\_routines.f90.

#### 7.41.2 Member Function Documentation

- 7.41.2.1 subroutine `COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_-`  
`INTERPOLATION_TYPE_SET::COORDINATE_SYSTEM_RADIAL_-`  
`INTERPOLATION_TYPE_SET_NUMBER` (INTEGER(INTG),intent(in)  
`USER_NUMBER`, INTEGER(INTG),intent(in) `RADIAL_INTERPOLATION_TYPE`,  
INTEGER(INTG),intent(out) `ERR`, TYPE(VARYING\_STRING),intent(out) `ERROR`, \*)  
[private]

Definition at line 1364 of file coordinate\_routines.f90.

- 7.41.2.2 subroutine `COORDINATE_ROUTINES::COORDINATE_SYSTEM_-`  
`RADIAL_INTERPOLATION_TYPE_SET::COORDINATE_-`  
`SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR`  
(TYPE(COORDINATE\_SYSTEM\_TYPE),pointer `COORDINATE_SYSTEM`,  
INTEGER(INTG),intent(in) `RADIAL_INTERPOLATION_TYPE`,  
INTEGER(INTG),intent(out) `ERR`, TYPE(VARYING\_STRING),intent(out) `ERROR`, \*)  
[private]

Definition at line 1395 of file coordinate\_routines.f90.

## **7.42 COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_TYPE\_SET Interface Reference**

### **Private Member Functions**

- subroutine [COORDINATE\\_SYSTEM\\_TYPE\\_SET\\_NUMBER](#) (*USER\_NUMBER*, *TYPE*, *ERR*, *ERROR,\**)
- subroutine [COORDINATE\\_SYSTEM\\_TYPE\\_SET\\_PTR](#) (*COORDINATE\_SYSTEM*, *TYPE*, *ERR*, *ERROR,\**)

#### **7.42.1 Detailed Description**

Definition at line 149 of file coordinate\_routines.f90.

#### **7.42.2 Member Function Documentation**

**7.42.2.1 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_TYPE\_SET::COORDINATE\_SYSTEM\_TYPE\_SET\_NUMBER** (*INTEGER(INTG),intent(in) USER\_NUMBER*, *INTEGER(INTG),intent(in) TYPE*, *INTEGER(INTG),intent(out) ERR*, *TYPE(VARYING\_STRING),intent(out) ERROR, \**) [private]

Definition at line 1471 of file coordinate\_routines.f90.

**7.42.2.2 subroutine COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_TYPE\_SET::COORDINATE\_SYSTEM\_TYPE\_SET\_PTR** (*TYPE(COORDINATE\_SYSTEM\_TYPE),pointer COORDINATE\_SYSTEM*, *INTEGER(INTG),intent(in) TYPE*, *INTEGER(INTG),intent(out) ERR*, *TYPE(VARYING\_STRING),intent(out) ERROR, \**) [private]

Definition at line 1502 of file coordinate\_routines.f90.

## 7.43 COORDINATE\_ROUTINES::COORDINATE\_SYSTEMS\_- TYPE Struct Reference

Collaboration diagram for COORDINATE\_ROUTINES::COORDINATE\_SYSTEMS\_TYPE:

### Private Attributes

- INTEGER(INTG) [NUMBER\\_OF\\_COORDINATE\\_SYSTEMS](#)
- TYPE([COORDINATE\\_SYSTEM\\_PTR\\_TYPE](#)), pointer [COORDINATE\\_SYSTEMS](#)

#### 7.43.1 Detailed Description

Definition at line 99 of file coordinate\_routines.f90.

#### 7.43.2 Member Data Documentation

##### 7.43.2.1 TYPE(COORDINATE\_SYSTEM\_PTR\_TYPE),pointer COORDINATE\_- ROUTINES::COORDINATE\_SYSTEMS\_TYPE::COORDINATE\_SYSTEMS [private]

Definition at line 101 of file coordinate\_routines.f90.

##### 7.43.2.2 INTEGER(INTG) COORDINATE\_ROUTINES::COORDINATE\_- SYSTEMS\_TYPE::NUMBER\_OF\_COORDINATE\_SYSTEMS [private]

Definition at line 100 of file coordinate\_routines.f90.

## 7.44 COORDINATE\_ROUTINES::D2ZX Interface Reference

### Private Member Functions

- REAL(DP) [D2ZX\\_DP](#) (COORDINATE\_SYSTEM, I, J, X, ERR, ERROR)

#### 7.44.1 Detailed Description

Definition at line 171 of file coordinate\_routines.f90.

#### 7.44.2 Member Function Documentation

7.44.2.1 REAL(DP) COORDINATE\_ROUTINES::D2ZX::D2ZX\_DP (TYPE(COORDINATE\_SYSTEM\_TYPE),intent(in) *COORDINATE\_SYSTEM*, INTEGER(INTG),intent(in) *I*, INTEGER(INTG),intent(in) *J*, REAL(DP),dimension(:),intent(in) *X*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*)  
[private]

Definition at line 2084 of file coordinate\_routines.f90.

## 7.45 COORDINATE\_ROUTINES::DXZ Interface Reference

### Private Member Functions

- REAL(DP) [DXZ\\_DP](#) (COORDINATE\_SYSTEM, I, X, ERR, ERROR)

#### 7.45.1 Detailed Description

Definition at line 165 of file coordinate\_routines.f90.

#### 7.45.2 Member Function Documentation

- 7.45.2.1 REAL(DP) COORDINATE\_ROUTINES::DXZ::DXZ\_DP (TYPE(COORDINATE\_SYSTEM\_TYPE),intent(in) *COORDINATE\_SYSTEM*, INTEGER(INTG),intent(in) *I*, REAL(DP),dimension(:),intent(in) *X*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]

Definition at line 1943 of file coordinate\_routines.f90.

## 7.46 COORDINATE\_ROUTINES::DZX Interface Reference

### Private Member Functions

- REAL(DP) [DZX\\_DP](#) (COORDINATE\_SYSTEM, I, X, ERR, ERROR)

#### 7.46.1 Detailed Description

Definition at line 177 of file coordinate\_routines.f90.

#### 7.46.2 Member Function Documentation

7.46.2.1 **REAL(DP) COORDINATE\_ROUTINES::DZX::DZX\_DP (TYPE(COORDINATE\_SYSTEM\_TYPE),intent(in) COORDINATE\_SYSTEM, INTEGER(INTG),intent(in) I, REAL(DP),dimension(:),intent(in) X, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

Definition at line 2353 of file coordinate\_routines.f90.

## 7.47 F90C::interface Interface Reference

### Public Member Functions

- subroutine [CSTRINGLEN](#) (LENGTH, CSTRING)
- subroutine [PACKCHARACTERS](#) (INTCHAR, POSITION, CSTRING)
- subroutine [UNPACKCHARACTERS](#) (INTCHAR, POSITION, CSTRING)

#### 7.47.1 Detailed Description

Definition at line 54 of file f90c\_f.f90.

#### 7.47.2 Member Function Documentation

##### 7.47.2.1 subroutine F90C::interface::CSTRINGLEN (INTEGER(INTG),intent(out) LENGTH, INTEGER(INTG),dimension(\*),intent(in) CSTRING)

Definition at line 56 of file f90c\_f.f90.

##### 7.47.2.2 subroutine F90C::interface::PACKCHARACTERS (INTE- GER(INTG),intent(in) INTCHAR, INTEGER(INTG),intent(in) POSITION, INTEGER(INTG),dimension(\*),intent(out) CSTRING)

Definition at line 63 of file f90c\_f.f90.

##### 7.47.2.3 subroutine F90C::interface::UNPACKCHARACTERS (INTE- GER(INTG),intent(out) INTCHAR, INTEGER(INTG),intent(in) POSITION, INTEGER(INTG),dimension(\*),intent(in) CSTRING)

Definition at line 70 of file f90c\_f.f90.

## 7.48 FIELD\_IO\_ROUTINES::FIELD\_IO\_ELEMENTALL\_INFO\_SET Struct Reference

contains information for parallel IO, and it is nodal base

Collaboration diagram for FIELD\_IO\_ROUTINES::FIELD\_IO\_ELEMENTALL\_INFO\_SET:

### Private Attributes

- TYPE(FIELDS\_TYPE), pointer FIELDS  
*A pointer to the fields defined on the region.*
- INTEGER(INTG) NUMBER\_OF\_ELEMENTS  
*Number of nodes in this computational node for NODAL\_INFO\_SET.*
- INTEGER(INTG), allocatable LIST\_OF\_GLOBAL\_NUMBER  
*the list of global numbering in each domain*
- TYPE(FIELD\_IO\_INFO\_SET), allocatable ELEMENTAL\_INFO\_SET  
*A list of nodal information for IO.*

### 7.48.1 Detailed Description

contains information for parallel IO, and it is nodal base

Definition at line 117 of file field\_IO\_routines.f90.

### 7.48.2 Member Data Documentation

#### 7.48.2.1 TYPE(FIELD\_IO\_INFO\_SET),allocatable FIELD\_IO\_ROUTINES::FIELD\_IO\_ELEMENTALL\_INFO\_SET::ELEMENTAL\_INFO\_SET [private]

A list of nodal information for IO.

Definition at line 122 of file field\_IO\_routines.f90.

#### 7.48.2.2 TYPE(FIELDS\_TYPE),pointer FIELD\_IO\_ROUTINES::FIELD\_IO\_ELEMENTALL\_INFO\_SET::FIELDS [private]

A pointer to the fields defined on the region.

Definition at line 118 of file field\_IO\_routines.f90.

#### 7.48.2.3 INTEGER(INTG),allocatable FIELD\_IO\_ROUTINES::FIELD\_IO\_ELEMENTALL\_INFO\_SET::LIST\_OF\_GLOBAL\_NUMBER [private]

the list of global numbering in each domain

Definition at line 121 of file field\_IO\_routines.f90.

**7.48.2.4 INTEGER(INTG) FIELD\_IO\_ROUTINES::FIELD\_IO\_ELEMENTALL\_INFO\_SET::NUMBER\_OF\_ELEMENTS [private]**

Number of nodes in this computational node for NODAL\_INFO\_SET.

Definition at line 119 of file field\_IO\_routines.f90.

## 7.49 FIELD\_IO\_ROUTINES::FIELD\_IO\_INFO\_SET Struct Reference

contains information for parallel IO, and it is nodal base

Collaboration diagram for FIELD\_IO\_ROUTINES::FIELD\_IO\_INFO\_SET:

### Private Attributes

- LOGICAL [SAME\\_HEADER](#)  
*determine whether we have same IO information as the previous one*
- INTEGER(INTG) [NUMBER\\_OF\\_COMPONENTS](#)  
*number of components in the component array, COMPONENT(:)*
- TYPE([FIELD\\_VARIABLE\\_COMPONENT\\_PTR\\_TYPE](#)), allocatable [COMPONENTS](#)  
*A array of pointers to those components of the node in this local domain.*

### 7.49.1 Detailed Description

contains information for parallel IO, and it is nodal base

Definition at line 98 of file field\_IO\_routines.f90.

### 7.49.2 Member Data Documentation

#### 7.49.2.1 TYPE(FIELD\_VARIABLE\_COMPONENT\_PTR\_TYPE),allocatable FIELD\_IO\_ROUTINES::FIELD\_IO\_INFO\_SET::COMPONENTS [private]

A array of pointers to those components of the node in this local domain.

Definition at line 104 of file field\_IO\_routines.f90.

#### 7.49.2.2 INTEGER(INTG) FIELD\_IO\_ROUTINES::FIELD\_IO\_INFO\_SET::NUMBER\_OF\_- COMPONENTS [private]

number of components in the component array, COMPONENT(:)

Definition at line 101 of file field\_IO\_routines.f90.

#### 7.49.2.3 LOGICAL FIELD\_IO\_ROUTINES::FIELD\_IO\_INFO\_SET::SAME\_HEADER [private]

determine whether we have same IO information as the previous one

Definition at line 100 of file field\_IO\_routines.f90.

## 7.50 FIELD\_IO\_ROUTINES::FIELD\_IO\_NODAL\_INFO\_SET Struct Reference

contains information for parallel IO, and it is nodal base

Collaboration diagram for FIELD\_IO\_ROUTINES::FIELD\_IO\_NODAL\_INFO\_SET:

### Private Attributes

- TYPE(FIELDS\_TYPE), pointer FIELDS  
*A pointe r to the fields defined on the region.*
- INTEGER(INTG) NUMBER\_OF\_NODES  
*Number of nodes in this computational node for NODAL\_INFO\_SET.*
- INTEGER(INTG), allocatable LIST\_OF\_GLOBAL\_NUMBER  
*the list of global numbering in each domain*
- TYPE(FIELD\_IO\_INFO\_SET), allocatable NODAL\_INFO\_SET  
*A list of nodal information for IO.*

### 7.50.1 Detailed Description

contains information for parallel IO, and it is nodal base

Definition at line 108 of file field\_IO\_routines.f90.

### 7.50.2 Member Data Documentation

#### 7.50.2.1 TYPE(FIELDS\_TYPE),pointer FIELD\_IO\_ROUTINES::FIELD\_IO\_NODAL\_INFO\_SET::FIELDS [private]

A pointe r to the fields defined on the region.

Definition at line 109 of file field\_IO\_routines.f90.

#### 7.50.2.2 INTEGER(INTG),allocatable FIELD\_IO\_ROUTINES::FIELD\_IO\_NODAL\_INFO\_SET::LIST\_OF\_GLOBAL\_NUMBER [private]

the list of global numbering in each domain

Definition at line 112 of file field\_IO\_routines.f90.

#### 7.50.2.3 TYPE(FIELD\_IO\_INFO\_SET),allocatable FIELD\_IO\_ROUTINES::FIELD\_IO\_NODAL\_INFO\_SET::NODAL\_INFO\_SET [private]

A list of nodal information for IO.

Definition at line 113 of file field\_IO\_routines.f90.

**7.50.2.4 INTEGER(INTG) FIELD\_IO\_ROUTINES::FIELD\_IO\_NODAL\_INFO\_SET::NUMBER\_OF\_NODES [private]**

Number of nodes in this computational node for NODAL\_INFO\_SET.

Definition at line 110 of file field\_IO\_routines.f90.

## 7.51 FIELD\_IO\_ROUTINES::FIELD\_VARIABLE\_COMPONENT\_PTR\_TYPE Struct Reference

field variable component type pointer for IO

Collaboration diagram for FIELD\_IO\_ROUTINES::FIELD\_VARIABLE\_COMPONENT\_PTR\_TYPE:

### Private Attributes

- TYPE(FIELD\_VARIABLE\_COMPONENT\_TYPE), pointer PTR  
*pointer field variable component*

#### 7.51.1 Detailed Description

field variable component type pointer for IO

Definition at line 93 of file field\_IO\_routines.f90.

#### 7.51.2 Member Data Documentation

##### 7.51.2.1 TYPE(FIELD\_VARIABLE\_COMPONENT\_TYPE),pointer FIELD\_IO\_ROUTINES::FIELD\_VARIABLE\_COMPONENT\_PTR\_TYPE::PTR [private]

pointer field variable component

Definition at line 94 of file field\_IO\_routines.f90.

## **7.52 FIELD\_IO\_ROUTINES::MESH\_ELEMENTS\_TYPE\_PTR\_TYPE Struct Reference**

field variable component type pointer for IO

Collaboration diagram for FIELD\_IO\_ROUTINES::MESH\_ELEMENTS\_TYPE\_PTR\_TYPE:

### **Private Attributes**

- TYPE(MESH\_ELEMENTS\_TYPE), pointer PTR  
*pointer field variable component*

#### **7.52.1 Detailed Description**

field variable component type pointer for IO

Definition at line 88 of file field\_IO\_routines.f90.

#### **7.52.2 Member Data Documentation**

##### **7.52.2.1 TYPE(MESH\_ELEMENTS\_TYPE),pointer FIELD\_IO\_ROUTINES::MESH\_ELEMENTS\_TYPE\_PTR\_TYPE::PTR [private]**

pointer field variable component

Definition at line 89 of file field\_IO\_routines.f90.

## 7.53 FIELD\_ROUTINES::FIELD\_COMPONENT\_- INTERPOLATION\_SET Interface Reference

### Private Member Functions

- [FIELD\\_COMPONENT\\_INTERPOLATION\\_SET\\_NUMBER](#)
- [FIELD\\_COMPONENT\\_INTERPOLATION\\_SET\\_PTR](#)

#### 7.53.1 Detailed Description

Definition at line 157 of file field\_routines.f90.

#### 7.53.2 Member Function Documentation

**7.53.2.1 FIELD\_ROUTINES::FIELD\_COMPONENT\_INTERPOLATION\_-  
SET::FIELD\_COMPONENT\_INTERPOLATION\_SET\_NUMBER ()**  
[private]

**7.53.2.2 FIELD\_ROUTINES::FIELD\_COMPONENT\_INTERPOLATION\_-  
SET::FIELD\_COMPONENT\_INTERPOLATION\_SET\_PTR ()**  
[private]

## **7.54 FIELD\_ROUTINES::FIELD\_COMPONENT\_MESH\_COMPONENT\_SET Interface Reference**

### **Private Member Functions**

- [FIELD\\_COMPONENT\\_MESH\\_COMPONENT\\_SET\\_NUMBER](#)
- [FIELD\\_COMPONENT\\_MESH\\_COMPONENT\\_SET\\_PTR](#)

#### **7.54.1 Detailed Description**

Definition at line 162 of file field\_routines.f90.

#### **7.54.2 Member Function Documentation**

**7.54.2.1 FIELD\_ROUTINES::FIELD\_COMPONENT\_MESH\_COMPONENT\_SET::FIELD\_COMPONENT\_MESH\_COMPONENT\_SET\_NUMBER ()**  
[private]

**7.54.2.2 FIELD\_ROUTINES::FIELD\_COMPONENT\_MESH\_COMPONENT\_SET::FIELD\_COMPONENT\_MESH\_COMPONENT\_SET\_PTR ()**  
[private]

## 7.55 FIELD\_ROUTINES::FIELD\_DEPENDENT\_TYPE\_SET Interface Reference

### Private Member Functions

- subroutine [FIELD\\_DEPENDENT\\_TYPE\\_SET\\_NUMBER](#) (*USER\_NUMBER*, *REGION*, *DEPENDENT\_TYPE*, *ERR*, *ERROR,\**)
- subroutine [FIELD\\_DEPENDENT\\_TYPE\\_SET\\_PTR](#) (*FIELD*, *DEPENDENT\_TYPE*, *ERR*, *ERROR,\**)

#### 7.55.1 Detailed Description

Definition at line 167 of file field\_routines.f90.

#### 7.55.2 Member Function Documentation

**7.55.2.1 subroutine FIELD\_ROUTINES::FIELD\_DEPENDENT\_TYPE\_SET::FIELD\_DEPENDENT\_TYPE\_SET\_NUMBER** (*INTEGER(INTG),intent(in)* *USER\_NUMBER*, *TYPE(REGION\_TYPE),pointer REGION*, *INTEGER(INTG),intent(in)* *DEPENDENT\_TYPE*, *INTEGER(INTG),intent(out)* *ERR*, *TYPE(VARYING\_STRING),intent(out)* *ERROR, \**) [private]

##### Parameters:

***USER\_NUMBER*** The user number of the field to set the dependent type for  
***REGION*** A pointer to the region containing the field

***DEPENDENT\_TYPE*** The dependent type to set/change for the field

See also:

[FIELD\\_ROUTINES::DependentTypes](#),[FIELD\\_ROUTINES](#)

***ERR*** The error code

***ERROR*** The error string

Definition at line 1183 of file field\_routines.f90.

**7.55.2.2 subroutine FIELD\_ROUTINES::FIELD\_DEPENDENT\_TYPE\_SET::FIELD\_DEPENDENT\_TYPE\_SET\_PTR** (*TYPE(FIELD\_TYPE),pointer FIELD*, *INTEGER(INTG),intent(in)* *DEPENDENT\_TYPE*, *INTEGER(INTG),intent(out)* *ERR*, *TYPE(VARYING\_STRING),intent(out)* *ERROR, \**) [private]

##### Parameters:

***FIELD*** A pointer to the field to set/change the dependent type for

***DEPENDENT\_TYPE*** The dependent type to set/change

See also:

[FIELD\\_ROUTINES::DependentTypes](#),[FIELD\\_ROUTINES](#)

***ERR*** The error code

***ERROR*** The error string

Definition at line 1214 of file field\_routines.f90.

## 7.56 FIELD\_ROUTINES::FIELD\_DIMENSION\_SET Interface Reference

### Private Member Functions

- subroutine [FIELD\\_DIMENSION\\_SET\\_NUMBER](#) (USER\_NUMBER, REGION, FIELD\_DIMENSION, ERR, ERROR,\*)
- subroutine [FIELD\\_DIMENSION\\_SET\\_PTR](#) (FIELD, FIELD\_DIMENSION, ERR, ERROR,\*)

#### 7.56.1 Detailed Description

Definition at line 172 of file field\_routines.f90.

#### 7.56.2 Member Function Documentation

##### 7.56.2.1 subroutine FIELD\_ROUTINES::FIELD\_DIMENSION\_SET::FIELD\_DIMENSION\_SET\_NUMBER (INTEGER(INTG),intent(in) *USER\_NUMBER*, TYPE(REGION\_TYPE),pointer *REGION*, INTEGER(INTG),intent(in) *FIELD\_DIMENSION*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

###### Parameters:

*USER\_NUMBER* The user number of the field to set the dimension for

*REGION* A pointer to the region containing the field

*FIELD\_DIMENSION* The dimension to set/change

See also:

[FIELD\\_ROUTINES::DimensionTypes](#),[FIELD\\_ROUTINES](#)

*ERR* The error code

*ERROR* The error string

Definition at line 1421 of file field\_routines.f90.

##### 7.56.2.2 subroutine FIELD\_ROUTINES::FIELD\_DIMENSION\_SET::FIELD\_DIMENSION\_SET\_PTR (TYPE(FIELD\_TYPE),pointer *FIELD*, INTEGER(INTG),intent(in) *FIELD\_DIMENSION*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

###### Parameters:

*FIELD* A pointer to the field to set/change the dimension for

*FIELD\_DIMENSION* The field dimension to set/change

See also:

[FIELD\\_ROUTINES::DimensionTypes](#),[FIELD\\_ROUTINES](#)

*ERR* The error code

*ERROR* The error string

Definition at line 1452 of file field\_routines.f90.

## 7.57 FIELD\_ROUTINES::FIELD\_GEOMETRIC\_FIELD\_SET Interface Reference

### Private Member Functions

- [FIELD\\_GEOMETRIC\\_FIELD\\_SET\\_NUMBER](#)
- [FIELD\\_GEOMETRIC\\_FIELD\\_SET\\_PTR](#)

#### 7.57.1 Detailed Description

Definition at line 177 of file field\_routines.f90.

#### 7.57.2 Member Function Documentation

**7.57.2.1 FIELD\_ROUTINES::FIELD\_GEOMETRIC\_FIELD\_SET::FIELD\_GEOMETRIC\_FIELD\_SET\_NUMBER () [private]**

**7.57.2.2 FIELD\_ROUTINES::FIELD\_GEOMETRIC\_FIELD\_SET::FIELD\_GEOMETRIC\_FIELD\_SET\_PTR () [private]**

## **7.58 FIELD\_ROUTINES::FIELD\_MESH\_DECOMPOSITION\_SET Interface Reference**

### **Private Member Functions**

- [FIELD\\_MESH\\_DECOMPOSITION\\_SET\\_NUMBER](#)
- [FIELD\\_MESH\\_DECOMPOSITION\\_SET\\_PTR](#)

#### **7.58.1 Detailed Description**

Definition at line 182 of file field\_routines.f90.

#### **7.58.2 Member Function Documentation**

**7.58.2.1 FIELD\_ROUTINES::FIELD\_MESH\_DECOMPOSITION\_SET::FIELD\_MESH\_DECOMPOSITION\_SET\_NUMBER () [private]**

**7.58.2.2 FIELD\_ROUTINES::FIELD\_MESH\_DECOMPOSITION\_SET::FIELD\_MESH\_DECOMPOSITION\_SET\_PTR () [private]**

## 7.59 FIELD\_ROUTINES::FIELD\_NUMBER\_OF\_COMPONENTS\_SET Interface Reference

### Private Member Functions

- [FIELD\\_NUMBER\\_OF\\_COMPONENTS\\_SET\\_NUMBER](#)
- [FIELD\\_NUMBER\\_OF\\_COMPONENTS\\_SET\\_PTR](#)

#### 7.59.1 Detailed Description

Definition at line 187 of file field\_routines.f90.

#### 7.59.2 Member Function Documentation

**7.59.2.1 FIELD\_ROUTINES::FIELD\_NUMBER\_OF\_COMPONENTS\_SET::FIELD\_NUMBER\_OF\_COMPONENTS\_SET\_NUMBER () [private]**

**7.59.2.2 FIELD\_ROUTINES::FIELD\_NUMBER\_OF\_COMPONENTS\_SET::FIELD\_NUMBER\_OF\_COMPONENTS\_SET\_PTR () [private]**

## **7.60 FIELD\_ROUTINES::FIELD\_NUMBER\_OF\_VARIABLES\_SET Interface Reference**

### **Private Member Functions**

- [FIELD\\_NUMBER\\_OF\\_VARIABLES\\_SET\\_NUMBER](#)
- [FIELD\\_NUMBER\\_OF\\_VARIABLES\\_SET\\_PTR](#)

#### **7.60.1 Detailed Description**

Definition at line 192 of file field\_routines.f90.

#### **7.60.2 Member Function Documentation**

**7.60.2.1 FIELD\_ROUTINES::FIELD\_NUMBER\_OF\_VARIABLES\_SET::FIELD\_NUMBER\_OF\_VARIABLES\_SET\_NUMBER () [private]**

**7.60.2.2 FIELD\_ROUTINES::FIELD\_NUMBER\_OF\_VARIABLES\_SET::FIELD\_NUMBER\_OF\_VARIABLES\_SET\_PTR () [private]**

## 7.61 FIELD\_ROUTINES::FIELD\_SCALING\_TYPE\_SET Interface Reference

### Private Member Functions

- FIELD\_SCALING\_TYPE\_SET\_NUMBER
- FIELD\_SCALING\_TYPE\_SET\_PTR

#### 7.61.1 Detailed Description

Definition at line 197 of file field\_routines.f90.

#### 7.61.2 Member Function Documentation

7.61.2.1 **FIELD\_ROUTINES::FIELD\_SCALING\_TYPE\_SET::FIELD\_SCALING\_TYPE\_SET\_-  
NUMBER () [private]**

7.61.2.2 **FIELD\_ROUTINES::FIELD\_SCALING\_TYPE\_SET::FIELD\_SCALING\_TYPE\_SET\_-  
PTR () [private]**

## 7.62 FIELD\_ROUTINES::FIELD\_TYPE\_SET Interface Reference

### Private Member Functions

- [FIELD\\_TYPE\\_SET\\_NUMBER](#)
- [FIELD\\_TYPE\\_SET\\_PTR](#)

#### 7.62.1 Detailed Description

Definition at line 202 of file field\_routines.f90.

#### 7.62.2 Member Function Documentation

**7.62.2.1 FIELD\_ROUTINES::FIELD\_TYPE\_SET::FIELD\_TYPE\_SET\_NUMBER ()**  
[private]

**7.62.2.2 FIELD\_ROUTINES::FIELD\_TYPE\_SET::FIELD\_TYPE\_SET\_PTR ()** [private]

## 7.63 GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_- BASIS\_SET Interface Reference

### Private Member Functions

- subroutine [GENERATED\\_MESH\\_BASIS\\_SET\\_NUMBER](#) (USER\_NUMBER, BASIS, ERR, ERROR,\*)
- subroutine [GENERATED\\_MESH\\_BASIS\\_SET\\_PTR](#) (GENERATED\_MESH, BASIS, ERR, ERROR,\*)

#### 7.63.1 Detailed Description

Definition at line 80 of file generated\_mesh\_routines.f90.

#### 7.63.2 Member Function Documentation

**7.63.2.1 subroutine GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_BASIS\_-  
SET::GENERATED\_MESH\_BASIS\_SET\_NUMBER (INTEGER(INTG),intent(in)  
USER\_NUMBER, TYPE(BASIS\_TYPE),pointer BASIS, INTEGER(INTG),intent(out)  
ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

Definition at line 165 of file generated\_mesh\_routines.f90.

**7.63.2.2 subroutine GENERATED\_MESH\_ROUTINES::GENERATED\_-  
MESH\_BASIS\_SET::GENERATED\_MESH\_BASIS\_SET\_PTR  
(TYPE(GENERATED\_MESH\_TYPE),pointer GENERATED\_MESH,  
TYPE(BASIS\_TYPE),pointer BASIS, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

#### Parameters:

**GENERATED\_MESH** A pointer to the generated mesh to set the basis of

**BASIS** The basis of mesh to generate

**See also:**

[GENERATED\\_MESH\\_ROUTINES\\_GeneratedMeshBasis](#),[GENERATED\\_MESH\\_-  
ROUTINES](#)

**ERR** The error code

**ERROR** The error string

Definition at line 196 of file generated\_mesh\_routines.f90.

## 7.64 GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_DESTROY Interface Reference

### Private Member Functions

- subroutine [GENERATED\\_MESH\\_DESTROY\\_NUMBER](#) (USER\_NUMBER, ERR, ERROR,\*)
- subroutine [GENERATED\\_MESH\\_DESTROY\\_PTR](#) (GENERATED\_MESH, ERR, ERROR,\*)

#### 7.64.1 Detailed Description

Definition at line 85 of file generated\_mesh\_routines.f90.

#### 7.64.2 Member Function Documentation

**7.64.2.1 subroutine GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_DESTROY::GENERATED\_MESH\_DESTROY\_NUMBER**  
**(INTEGER(INTG),intent(in) *USER\_NUMBER*, INTEGER(INTG),intent(out) *ERR*,**  
**TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

##### Parameters:

***USER\_NUMBER*** The user number of generated mesh to destroy

***ERR*** The error code

***ERROR*** The error string

Definition at line 353 of file generated\_mesh\_routines.f90.

**7.64.2.2 subroutine GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_DESTROY::GENERATED\_MESH\_DESTROY\_PTR**  
**(TYPE(GENERATED\_MESH\_TYPE),pointer *GENERATED\_MESH*,**  
**INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**  
**[private]**

##### Parameters:

***GENERATED\_MESH*** A pointer to the generated mesh to destroy

***ERR*** The error code

***ERROR*** The error string

Definition at line 400 of file generated\_mesh\_routines.f90.

## 7.65 GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_- EXTENT\_SET Interface Reference

### Private Member Functions

- subroutine [GENERATED\\_MESH\\_EXTENT\\_SET\\_NUMBER](#) (USER\_NUMBER, MAX\_EXTENT, ERR, ERROR,\*)
- subroutine [GENERATED\\_MESH\\_EXTENT\\_SET\\_PTR](#) (GENERATED\_MESH, MAX\_EXTENT, ERR, ERROR,\*)

#### 7.65.1 Detailed Description

Definition at line 90 of file generated\_mesh\_routines.f90.

#### 7.65.2 Member Function Documentation

7.65.2.1 subroutine **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_EXTENT\_SET::GENERATED\_MESH\_EXTENT\_SET\_NUMBER** (INTEGER(INTG),intent(in) *USER\_NUMBER*, REAL(DP),dimension(:),intent(in) *MAX\_EXTENT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 493 of file generated\_mesh\_routines.f90.

7.65.2.2 subroutine **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_EXTENT\_SET::GENERATED\_MESH\_EXTENT\_SET\_PTR** (TYPE(GENERATED\_MESH\_TYPE),pointer *GENERATED\_MESH*, REAL(DP),dimension(:),intent(in) *MAX\_EXTENT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

#### Parameters:

**GENERATED\_MESH** A pointer to the generated mesh to set the type of

**ERR** The error code

**ERROR** The error string

Definition at line 524 of file generated\_mesh\_routines.f90.

## 7.66

GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_NUMBER\_OF\_ELEMENTS\_SET  
Interface Reference  
**7.66 GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_-<sup>865</sup>**  
**NUMBER\_OF\_ELEMENTS\_SET Interface Reference**

### Private Member Functions

- subroutine **GENERATED\_MESH\_NUMBER\_OF\_ELEMENTS\_SET\_NUMBER** (USER\_-  
NUMBER, NUMBER\_OF\_ELEMENTS\_XI, ERR, ERROR,\*)
- subroutine **GENERATED\_MESH\_NUMBER\_OF\_ELEMENTS\_SET\_PTR** (GENERATED\_-  
MESH, NUMBER\_OF\_ELEMENTS\_XI, ERR, ERROR,\*)

#### 7.66.1 Detailed Description

Definition at line 95 of file generated\_mesh\_routines.f90.

#### 7.66.2 Member Function Documentation

**7.66.2.1 subroutine GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_-  
NUMBER\_OF\_ELEMENTS\_SET::GENERATED\_MESH\_NUMBER\_OF\_-  
ELEMENTS\_SET\_NUMBER (INTEGER(INTG),intent(in) *USER\_NUMBER*,  
INTEGER(INTG),dimension(:),intent(in) *NUMBER\_OF\_ELEMENTS\_XI*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)  
[private]**

Definition at line 667 of file generated\_mesh\_routines.f90.

**7.66.2.2 subroutine GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_NUMBER\_-  
OF\_ELEMENTS\_SET::GENERATED\_MESH\_NUMBER\_OF\_ELEMENTS\_SET\_PTR  
(TYPE(GENERATED\_MESH\_TYPE),pointer *GENERATED\_MESH*,  
INTEGER(INTG),dimension(:),intent(in) *NUMBER\_OF\_ELEMENTS\_XI*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)  
[private]**

#### Parameters:

**GENERATED\_MESH** A pointer to the generated mesh to set the type of  
**ERR** The error code  
**ERROR** The error string

Definition at line 699 of file generated\_mesh\_routines.f90.

## 7.67 GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_- ORIGIN\_SET Interface Reference

### Private Member Functions

- subroutine `GENERATED_MESH_ORIGIN_SET_NUMBER` (`USER_NUMBER`, `ORIGIN`, `ERR`, `ERROR,*`)
- subroutine `GENERATED_MESH_ORIGIN_SET_PTR` (`GENERATED_MESH`, `ORIGIN`, `ERR`, `ERROR,*`)

#### 7.67.1 Detailed Description

Definition at line 100 of file generated\_mesh\_routines.f90.

#### 7.67.2 Member Function Documentation

**7.67.2.1 subroutine `GENERATED_MESH_ROUTINES::GENERATED_-  
MESH_ORIGIN_SET::GENERATED_MESH_ORIGIN_SET_NUMBER`**  
`(INTEGER(INTG),intent(in) USER_NUMBER, REAL(DP),dimension(:),intent(in)  
ORIGIN, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)  
ERROR, *) [private]`

Definition at line 779 of file generated\_mesh\_routines.f90.

**7.67.2.2 subroutine `GENERATED_MESH_ROUTINES::GENERATED_-  
MESH_ORIGIN_SET::GENERATED_MESH_ORIGIN_SET_PTR`**  
`(TYPE(GENERATED_MESH_TYPE),pointer GENERATED_MESH,  
REAL(DP),dimension(:),intent(in) ORIGIN, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

#### Parameters:

**`GENERATED_MESH`** A pointer to the generated mesh to set the type of

**`ERR`** The error code

**`ERROR`** The error string

Definition at line 810 of file generated\_mesh\_routines.f90.

## **7.68 GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_TYPE\_SET Interface Reference**

### **Private Member Functions**

- subroutine [GENERATED\\_MESH\\_TYPE\\_SET\\_NUMBER](#) (USER\_NUMBER, GENERATED\_TYPE, ERR, ERROR,\*)
- subroutine [GENERATED\\_MESH\\_TYPE\\_SET\\_PTR](#) (GENERATED\_MESH, GENERATED\_TYPE, ERR, ERROR,\*)

#### **7.68.1 Detailed Description**

Definition at line 105 of file generated\_mesh\_routines.f90.

#### **7.68.2 Member Function Documentation**

**7.68.2.1 subroutine GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_TYPE\_SET::GENERATED\_MESH\_TYPE\_SET\_NUMBER** (INTEGER(INTG),intent(in) *USER\_NUMBER*, INTEGER(INTG),intent(in) *GENERATED\_TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 1131 of file generated\_mesh\_routines.f90.

**7.68.2.2 subroutine GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_TYPE\_SET::GENERATED\_MESH\_TYPE\_SET\_PTR** (TYPE(GENERATED\_MESH\_TYPE),pointer *GENERATED\_MESH*, INTEGER(INTG),intent(in) *GENERATED\_TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

#### **Parameters:**

***GENERATED\_MESH*** A pointer to the generated mesh to set the type of

***GENERATED\_TYPE*** The type of mesh to generate

**See also:**

[GENERATED\\_MESH\\_ROUTINES::GeneratedMeshTypes](#),[GENERATED\\_MESH\\_ROUTINES](#)

***ERR*** The error code

***ERROR*** The error string

Definition at line 1162 of file generated\_mesh\_routines.f90.

## 7.69 INPUT\_OUTPUT::WRITE\_STRING Interface Reference

Write a string to a given output stream.

### Private Member Functions

- subroutine [WRITE\\_STRING\\_C](#) (ID, STRING, ERR, ERROR,\*)
- subroutine [WRITE\\_STRING\\_VS](#) (ID, STRING, ERR, ERROR,\*)

#### 7.69.1 Detailed Description

Write a string to a given output stream.

Definition at line 71 of file input\_output.f90.

#### 7.69.2 Member Function Documentation

**7.69.2.1 subroutine INPUT\_OUTPUT::WRITE\_STRING::WRITE\_STRING\_C**  
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) STRING,  
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`  
 [private]

##### Parameters:

**ID** The ID of the output stream to write to

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

**STRING** The string to write

**ERR** The error code

**ERROR** The error string

Definition at line 221 of file input\_output.f90.

**7.69.2.2 subroutine INPUT\_OUTPUT::WRITE\_STRING::WRITE\_STRING\_VS**  
`(INTEGER(INTG),intent(in) ID, TYPE(VARYING_STRING),intent(in) STRING,  
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`  
 [private]

##### Parameters:

**ID** The ID of the output stream to write to

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

**STRING** The string to write

**ERR** The error code

**ERROR** The error string

Definition at line 247 of file input\_output.f90.

## 7.70 INPUT\_OUTPUT::WRITE\_STRING\_FMT\_TWO\_VALUE Interface Reference

Write a string, value, string then a value with the values formatted in a particular way to a given output stream.

### Private Member Functions

- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_C\\_C](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_C\\_DP](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_C\\_INTG](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_C\\_L](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_C\\_SP](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_C\\_VS](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_DP\\_C](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_DP\\_DP](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_DP\\_INTG](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_DP\\_L](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_DP\\_SP](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_DP\\_VS](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_INTG\\_C](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_INTG\\_DP](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_INTG\\_INTG](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_INTG\\_L](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_INTG\\_SP](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_INTG\\_VS](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_L\\_C](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_L\\_DP](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_L\\_INTG](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_L\\_L](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_L\\_SP](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_L\\_VS](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_SP\\_C](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_SP\\_DP](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_SP\\_INTG](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_SP\\_L](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_SP\\_SP](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_SP\\_VS](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_VS\\_C](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_VS\\_DP](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_VS\\_INTG](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_VS\\_L](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_VS\\_SP](#)
- [WRITE\\_STRING\\_FMT\\_TWO\\_VALUE\\_VS\\_VS](#)

#### 7.70.1 Detailed Description

Write a string, value, string then a value with the values formatted in a particular way to a given output stream.

Definition at line 139 of file input\_output.f90.



## 7.70.2 Member Function Documentation

- 7.70.2.1 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-  
TWO_VALUE_C_C()` [private]
- 7.70.2.2 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-  
TWO_VALUE_C_DP()` [private]
- 7.70.2.3 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-  
TWO_VALUE_C_INTG()` [private]
- 7.70.2.4 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-  
TWO_VALUE_C_L()` [private]
- 7.70.2.5 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-  
TWO_VALUE_C_SP()` [private]
- 7.70.2.6 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-  
TWO_VALUE_C_VS()` [private]
- 7.70.2.7 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-  
TWO_VALUE_DP_C()` [private]
- 7.70.2.8 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-  
TWO_VALUE_DP_DP()` [private]
- 7.70.2.9 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-  
TWO_VALUE_DP_INTG()` [private]
- 7.70.2.10 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-  
TWO_VALUE_DP_L()` [private]
- 7.70.2.11 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-  
TWO_VALUE_DP_SP()` [private]
- 7.70.2.12 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-  
TWO_VALUE_DP_VS()` [private]
- 7.70.2.13 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-  
TWO_VALUE_INTG_C()` [private]
- 7.70.2.14 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-  
TWO_VALUE_INTG_DP()` [private]
- 7.70.2.15 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-  
TWO_VALUE_INTG_INTG()` [private]
- 7.70.2.16 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-  
TWO_VALUE_INTG_L()` [private]
- 7.70.2.17 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-  
TWO_VALUE_INTG_SP()` [private]
- 7.70.2.18 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-  
TWO_VALUE_INTG_VS()` [private]
- 7.70.2.19 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-  
TWO_VALUE_L_C()` [private]
- 7.70.2.20 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-  
TWO_VALUE_L_DP()` [private]

## 7.71 INPUT\_OUTPUT::WRITE\_STRING\_FMT\_VALUE Interface Reference

Write a string followed by a value formatted in a particular way to a specified output stream.

### Private Member Functions

- subroutine [WRITE\\_STRING\\_FMT\\_VALUE\\_C](#) (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)
- subroutine [WRITE\\_STRING\\_FMT\\_VALUE\\_DP](#) (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)
- subroutine [WRITE\\_STRING\\_FMT\\_VALUE\\_INTG](#) (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)
- subroutine [WRITE\\_STRING\\_FMT\\_VALUE\\_LINTG](#) (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)
- subroutine [WRITE\\_STRING\\_FMT\\_VALUE\\_L](#) (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)
- subroutine [WRITE\\_STRING\\_FMT\\_VALUE\\_SP](#) (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)
- subroutine [WRITE\\_STRING\\_FMT\\_VALUE\\_VS](#) (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)

#### 7.71.1 Detailed Description

Write a string followed by a value formatted in a particular way to a specified output stream.

Definition at line 128 of file input\_output.f90.

#### 7.71.2 Member Function Documentation

**7.71.2.1 subroutine INPUT\_OUTPUT::WRITE\_STRING\_FMT\_VALUE::WRITE\_STRING\_FMT\_VALUE\_C (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, CHARACTER(LEN=\*),intent(in) *VALUE*, CHARACTER(LEN=\*),intent(in) *FORMAT\_STRING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

##### Parameters:

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***VALUE*** The value to be output

***FORMAT\_STRING*** The format string to be used to format the value

***ERR*** The error code

***ERROR*** The error string

Definition at line 1680 of file input\_output.f90.

---

**7.71.2.2 subroutine INPUT\_OUTPUT::WRITE\_STRING\_FMT\_VALUE::WRITE\_STRING\_FMT\_VALUE\_DP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, REAL(DP),intent(in) *VALUE*, CHARACTER(LEN=\*),intent(in) *FORMAT\_STRING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*VALUE* The value to be output

*FORMAT\_STRING* The format string to be used to format the value

*ERR* The error code

*ERROR* The error string

Definition at line 1710 of file input\_output.f90.

**7.71.2.3 subroutine INPUT\_OUTPUT::WRITE\_STRING\_FMT\_VALUE::WRITE\_STRING\_FMT\_VALUE\_INTG (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, INTEGER(INTG),intent(in) *VALUE*, CHARACTER(LEN=\*),intent(in) *FORMAT\_STRING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*VALUE* The value to be output

*FORMAT\_STRING* The format string to be used to format the value

*ERR* The error code

*ERROR* The error string

Definition at line 1741 of file input\_output.f90.

**7.71.2.4 subroutine INPUT\_OUTPUT::WRITE\_STRING\_FMT\_VALUE::WRITE\_STRING\_FMT\_VALUE\_L (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, LOGICAL,intent(in) *VALUE*, CHARACTER(LEN=\*),intent(in) *FORMAT\_STRING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**VALUE** The value to be output

**FORMAT\_STRING** The format string to be used to format the value

**ERR** The error code

**ERROR** The error string

Definition at line 1801 of file input\_output.f90.

**7.71.2.5 subroutine INPUT\_OUTPUT::WRITE\_STRING\_FMT\_VALUE::WRITE\_STRING\_FMT\_VALUE\_LINTG (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, INTEGER(LINTG),intent(in) *VALUE*, CHARACTER(LEN=\*),intent(in) *FORMAT\_STRING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

#### Parameters:

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**VALUE** The value to be output

**FORMAT\_STRING** The format string to be used to format the value

**ERR** The error code

**ERROR** The error string

Definition at line 1771 of file input\_output.f90.

**7.71.2.6 subroutine INPUT\_OUTPUT::WRITE\_STRING\_FMT\_VALUE::WRITE\_STRING\_FMT\_VALUE\_SP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, REAL(SP),intent(in) *VALUE*, CHARACTER(LEN=\*),intent(in) *FORMAT\_STRING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

#### Parameters:

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**VALUE** The value to be output

**FORMAT\_STRING** The format string to be used to format the value

**ERR** The error code

**ERROR** The error string

Definition at line 1831 of file input\_output.f90.

**7.71.2.7 subroutine INPUT\_OUTPUT::WRITE\_STRING\_FMT\_VALUE::WRITE\_STRING\_FMT\_VALUE\_VS (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, TYPE(VARYING\_STRING),intent(in) *VALUE*, CHARACTER(LEN=\*),intent(in) *FORMAT\_STRING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*VALUE* The value to be output

*FORMAT\_STRING* The format string to be used to format the value

*ERR* The error code

*ERROR* The error string

Definition at line 1862 of file input\_output.f90.

## 7.72 INPUT\_OUTPUT::WRITE\_STRING\_IDX\_VECTOR Interface Reference

Write a string followed by a indexed vector to a specified output stream.

### Private Member Functions

- [WRITE\\_STRING\\_IDX\\_VECTOR\\_DP](#)
- [WRITE\\_STRING\\_IDX\\_VECTOR\\_INTG](#)
- [WRITE\\_STRING\\_IDX\\_VECTOR\\_LINTG](#)
- [WRITE\\_STRING\\_IDX\\_VECTOR\\_L](#)
- [WRITE\\_STRING\\_IDX\\_VECTOR\\_SP](#)

#### 7.72.1 Detailed Description

Write a string followed by a indexed vector to a specified output stream.

Definition at line 188 of file input\_output.f90.

#### 7.72.2 Member Function Documentation

- 7.72.2.1 [INPUT\\_OUTPUT::WRITE\\_STRING\\_IDX\\_VECTOR::WRITE\\_STRING\\_IDX\\_VECTOR\\_DP \(\)](#) [private]
- 7.72.2.2 [INPUT\\_OUTPUT::WRITE\\_STRING\\_IDX\\_VECTOR::WRITE\\_STRING\\_IDX\\_VECTOR\\_INTG \(\)](#) [private]
- 7.72.2.3 [INPUT\\_OUTPUT::WRITE\\_STRING\\_IDX\\_VECTOR::WRITE\\_STRING\\_IDX\\_VECTOR\\_L \(\)](#) [private]
- 7.72.2.4 [INPUT\\_OUTPUT::WRITE\\_STRING\\_IDX\\_VECTOR::WRITE\\_STRING\\_IDX\\_VECTOR\\_LINTG \(\)](#) [private]
- 7.72.2.5 [INPUT\\_OUTPUT::WRITE\\_STRING\\_IDX\\_VECTOR::WRITE\\_STRING\\_IDX\\_VECTOR\\_SP \(\)](#) [private]

## 7.73 INPUT\_OUTPUT::WRITE\_STRING\_MATRIX Interface Reference

Write a string followed by a matrix to a specified output stream.

### Private Member Functions

- subroutine `WRITE_STRING_MATRIX_DP` (*ID*, *FIRST\_ROW*, *DELTA\_ROW*, *LAST\_ROW*, *FIRST\_COLUMN*, *DELTA\_COLUMN*, *LAST\_COLUMN*, *NUMBER\_FIRS*,&*NUMBER\_REPEAT*, *MATRIX*, *INDEX\_FORMAT\_TYPE*, *MATRIX\_NAME\_FORMAT*, *ROW\_INDEX\_FORMAT*, *FIRST\_FORMAT*, *REPEAT\_FORMAT*, *ERR*, *ERROR*,\*)
- subroutine `WRITE_STRING_MATRIX_INTG` (*ID*, *FIRST\_ROW*, *DELTA\_ROW*, *LAST\_ROW*, *FIRST\_COLUMN*, *DELTA\_COLUMN*, *LAST\_COLUMN*, *NUMBER\_FI ST*,&*NUMBER\_REPEAT*, *MATRIX*, *INDEX\_FORMAT\_TYPE*, *MATRIX\_NAME\_FORMAT*, *ROW\_INDEX\_FORMAT*, *FIRST\_FORMAT*, *REPEAT\_FORMAT*, *ERR*, *ERROR*,\*)
- subroutine `WRITE_STRING_MATRIX_LINTG` (*ID*, *FIRST\_ROW*, *DELTA\_ROW*, *LAST\_ROW*, *FIRST\_COLUMN*, *DELTA\_COLUMN*, *LAST\_COLUMN*, *NUMBER\_F RST*,&*NUMBER\_REPEAT*, *MATRIX*, *INDEX\_FORMAT\_TYPE*, *MATRIX\_NAME\_FORMAT*, *ROW\_INDEX\_FORMAT*, *FIRST\_FORMAT*, *REPEAT\_FORMAT*, *ERR*, *ERROR*,\*)
- subroutine `WRITE_STRING_MATRIX_L` (*ID*, *FIRST\_ROW*, *DELTA\_ROW*, *LAST\_ROW*, *FIRST\_COLUMN*, *DELTA\_COLUMN*, *LAST\_COLUMN*, *NUMBER\_FIRST* &*NUMBER\_REPEAT*, *MATRIX*, *INDEX\_FORMAT\_TYPE*, *MATRIX\_NAME\_FORMAT*, *ROW\_INDEX\_FORMAT*, *FIRST\_FORMAT*, *REPEAT\_FORMAT*, *ERR*, *ERROR*,\*)
- subroutine `WRITE_STRING_MATRIX_SP` (*ID*, *FIRST\_ROW*, *DELTA\_ROW*, *LAST\_ROW*, *FIRST\_COLUMN*, *DELTA\_COLUMN*, *LAST\_COLUMN*, *NUMBER\_FIRS*,&*NUMBER\_REPEAT*, *MATRIX*, *INDEX\_FORMAT\_TYPE*, *MATRIX\_NAME\_FORMAT*, *ROW\_INDEX\_FORMAT*, *FIRST\_FORMAT*, *REPEAT\_FORMAT*, *ERR*, *ERROR*,\*)

### 7.73.1 Detailed Description

Write a string followed by a matrix to a specified output stream.

Definition at line 197 of file input\_output.f90.

### 7.73.2 Member Function Documentation

**7.73.2.1 subroutine INPUT\_OUTPUT::WRITE\_STRING\_MATRIX::WRITE\_STRING\_-  
MATRIX\_DP (INTEGER(INTG),intent(in) *ID*, INTEGER(INTG),intent(in)  
*FIRST\_ROW*, INTEGER(INTG),intent(in) *DELTA\_ROW*, INTEGER(INTG),intent(in)  
*LAST\_ROW*, INTEGER(INTG),intent(in) *FIRST\_COLUMN*,  
INTEGER(INTG),intent(in) *DELTA\_COLUMN*, INTEGER(INTG),intent(in)  
*LAST\_COLUMN*, *NUMBER\_FIRS*, &,intent(in) *NUMBER\_REPEAT*,  
INTEGER(INTG),dimension(:, :, intent(in) *matrix*, INTEGER(INTG),intent(in)  
*INDEX\_FORMAT\_TYPE*, CHARACTER(LEN=\*=),intent(in)  
*MATRIX\_NAME\_FORMAT*, CHARACTER(LEN=\*=),intent(in)  
*ROW\_INDEX\_FORMAT*, CHARACTER(LEN=\*=),intent(in) *FIRST\_FORMAT*,  
CHARACTER(LEN=\*=),intent(in) *REPEAT\_FORMAT*, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

#### Parameters:

*ID* The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

**FIRST\_ROW** The first row of the matrix to be output

**DELTA\_ROW** The delta row increment to be used when outputting the first through to the last matrix row

**LAST\_ROW** The last row of the matrix to be output

**FIRST\_COLUMN** The first column of the matrix to be output

**DELTA\_COLUMN** The delta column increase to be used when outputting the first through to the last matrix column

**LAST\_COLUMN** The last column of the matrix to be output

**INDEX\_FORMAT\_TYPE** The format type to be used for the matrix name and indices

See also:

[INPUT\\_OUTPUT::MatrixNameIndexFormat](#),[INPUT\\_OUTPUT::MatrixNameIndexFormat](#)

**MATRIX\_NAME\_FORMAT** The format string to be used to format the matrix name

**ROW\_INDEX\_FORMAT** The format string to be used to format the row indices

**FIRST\_FORMAT** The format string to be used for the first line of output

**REPEAT\_FORMAT** The format type to be used for the second and subsequently repeated lines of output

**ERR** The error code

**ERROR** The error string

Definition at line 3655 of file input\_output.f90.

**7.73.2.2 subroutine INPUT\_OUTPUT::WRITE\_STRING\_MATRIX::WRITE\_STRING\_-  
MATRIX\_INTG (INTEGER(INTG),intent(in) *ID*, INTEGER(INTG),intent(in)  
FIRST\_ROW, INTEGER(INTG),intent(in) DELTA\_ROW, INTEGER(INTG),intent(in)  
LAST\_ROW, INTEGER(INTG),intent(in) FIRST\_COLUMN,  
INTEGER(INTG),intent(in) DELTA\_COLUMN, INTEGER(INTG),intent(in)  
LAST\_COLUMN, NUMBER\_FI ST, &,intent(in) NUMBER\_REPEAT,  
INTEGER(INTG),dimension(:, :, intent(in) *matrix*, INTEGER(INTG),intent(in)  
INDEX\_FORMAT\_TYPE, CHARACTER(LEN=\*),intent(in)  
MATRIX\_NAME\_FORMAT, CHARACTER(LEN=\*),intent(in)  
ROW\_INDEX\_FORMAT, CHARACTER(LEN=\*),intent(in) FIRST\_FORMAT,  
CHARACTER(LEN=\*),intent(in) REPEAT\_FORMAT, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

**FIRST\_ROW** The first row of the matrix to be output

**DELTA\_ROW** The delta row increment to be used when outputting the first through to the last matrix row

**LAST\_ROW** The last row of the matrix to be output

**FIRST\_COLUMN** The first column of the matrix to be output

**DELTA\_COLUMN** The delta column increase to be used when outputing the first through to the last matrix column

**LAST\_COLUMN** The last column of the matrix to be output

**INDEX\_FORMAT\_TYPE** The format type to be used for the matrix name and indices

See also:

[INPUT\\_OUTPUT::MatrixNameIndexFormat](#),[INPUT\\_OUTPUT::MatrixNameIndexFormat](#)

**MATRIX\_NAME\_FORMAT** The format string to be used to format the matrix name

**ROW\_INDEX\_FORMAT** The format string to be used to format the row indices

**FIRST\_FORMAT** The format string to be used for the first line of output

**REPEAT\_FORMAT** The format type to be used for the second and subsequently repeated lines of output

**ERR** The error code

**ERROR** The error string

Definition at line 3720 of file input\_output.f90.

**7.73.2.3 subroutine INPUT\_OUTPUT::WRITE\_STRING\_MATRIX::WRITE\_STRING\_-  
MATRIX\_L (INTEGER(INTG),intent(in) *ID*, INTEGER(INTG),intent(in) *FIRST\_ROW*,  
INTEGER(INTG),intent(in) *DELTA\_ROW*, INTEGER(INTG),intent(in) *LAST\_ROW*,  
INTEGER(INTG),intent(in) *FIRST\_COLUMN*, INTEGER(INTG),intent(in)  
*DELTA\_COLUMN*, INTEGER(INTG),intent(in) *LAST\_COLUMN*, NUMBER\_FIRST  
&,intent(in) *NUMBER\_REPEAT*, INTEGER(INTG),dimension(:, :, :),intent(in) *matrix*,  
INTEGER(INTG),intent(in) *INDEX\_FORMAT\_TYPE*, CHARACTER(LEN=\*=\*),intent(in)  
*MATRIX\_NAME\_FORMAT*, CHARACTER(LEN=\*=\*),intent(in) *ROW\_INDEX\_-  
FORMAT*, CHARACTER(LEN=\*=\*),intent(in) *FIRST\_FORMAT*,  
CHARACTER(LEN=\*=\*),intent(in) *REPEAT\_FORMAT*, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \* ) [private]**

Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

**FIRST\_ROW** The first row of the matrix to be output

**DELTA\_ROW** The delta row increment to be used when outputing the first through to the last matrix row

**LAST\_ROW** The last row of the matrix to be output

**FIRST\_COLUMN** The first column of the matrix to be output

**DELTA\_COLUMN** The delta column increase to be used when outputing the first through to the last matrix column

**LAST\_COLUMN** The last column of the matrix to be output

**INDEX\_FORMAT\_TYPE** The format type to be used for the matrix name and indices

See also:

[INPUT\\_OUTPUT::MatrixNameIndexFormat](#),[INPUT\\_OUTPUT::MatrixNameIndexFormat](#)

**MATRIX\_NAME\_FORMAT** The format string to be used to format the matrix name

**ROW\_INDEX\_FORMAT** The format string to be used to format the row indices

**FIRST\_FORMAT** The format string to be used for the first line of output

**REPEAT\_FORMAT** The format type to be used for the second and subsequently repeated lines of output

**ERR** The error code

**ERROR** The error string

Definition at line 3857 of file input\_output.f90.

**7.73.2.4 subroutine INPUT\_OUTPUT::WRITE\_STRING\_MATRIX::WRITE\_STRING\_-  
MATRIX\_LINTG (INTEGER(INTG),intent(in) *ID*, INTEGER(INTG),intent(in)  
*FIRST\_ROW*, INTEGER(INTG),intent(in) *DELTA\_ROW*, INTEGER(INTG),intent(in)  
*LAST\_ROW*, INTEGER(INTG),intent(in) *FIRST\_COLUMN*,  
INTEGER(INTG),intent(in) *DELTA\_COLUMN*, INTEGER(INTG),intent(in)  
*LAST\_COLUMN*, NUMBER\_F *RST*, &,intent(in) *NUMBER\_REPEAT*,  
INTEGER(INTG),dimension(:, :, intent(in) *matrix*, INTEGER(INTG),intent(in)  
*INDEX\_FORMAT\_TYPE*, CHARACTER(LEN=\*=),intent(in)  
*MATRIX\_NAME\_FORMAT*, CHARACTER(LEN=\*=),intent(in)  
*ROW\_INDEX\_FORMAT*, CHARACTER(LEN=\*=),intent(in) *FIRST\_FORMAT*,  
CHARACTER(LEN=\*=),intent(in) *REPEAT\_FORMAT*, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

#### Parameters:

**ID** The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_ROW** The first row of the matrix to be output

**DELTA\_ROW** The delta row increment to be used when outputing the first through to the last matrix row

**LAST\_ROW** The last row of the matrix to be output

**FIRST\_COLUMN** The first column of the matrix to be output

**DELTA\_COLUMN** The delta column increase to be used when outputing the first through to the last matrix column

**LAST\_COLUMN** The last column of the matrix to be output

**INDEX\_FORMAT\_TYPE** The format type to be used for the matrix name and indices

See also:

[INPUT\\_OUTPUT::MatrixNameIndexFormat](#), [INPUT\\_OUTPUT::MatrixNameIndexFormat](#)

**MATRIX\_NAME\_FORMAT** The format string to be used to format the matrix name

**ROW\_INDEX\_FORMAT** The format string to be used to format the row indices

**FIRST\_FORMAT** The format string to be used for the first line of output

**REPEAT\_FORMAT** The format type to be used for the second and subsequently repeated lines of output

**ERR** The error code

**ERROR** The error string

Definition at line 3792 of file input\_output.f90.

---

**7.73.2.5 subroutine INPUT\_OUTPUT::WRITE\_STRING\_MATRIX::WRITE\_STRING\_-  
MATRIX\_SP (INTEGER(INTG),intent(in) *ID*, INTEGER(INTG),intent(in)  
*FIRST\_ROW*, INTEGER(INTG),intent(in) *DELTA\_ROW*, INTEGER(INTG),intent(in)  
*LAST\_ROW*, INTEGER(INTG),intent(in) *FIRST\_COLUMN*,  
INTEGER(INTG),intent(in) *DELTA\_COLUMN*, INTEGER(INTG),intent(in)  
*LAST\_COLUMN*, NUMBER\_FIRS, &,intent(in) *NUMBER\_REPEAT*,  
INTEGER(INTG),dimension(:, :, intent(in) *matrix*, INTEGER(INTG),intent(in)  
*INDEX\_FORMAT\_TYPE*, CHARACTER(LEN=\*=),intent(in)  
*MATRIX\_NAME\_FORMAT*, CHARACTER(LEN=\*=),intent(in)  
*ROW\_INDEX\_FORMAT*, CHARACTER(LEN=\*=),intent(in) *FIRST\_FORMAT*,  
CHARACTER(LEN=\*=),intent(in) *REPEAT\_FORMAT*, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_ROW*** The first row of the matrix to be output

***DELTA\_ROW*** The delta row increment to be used when outputting the first through to the last matrix row

***LAST\_ROW*** The last row of the matrix to be output

***FIRST\_COLUMN*** The first column of the matrix to be output

***DELTA\_COLUMN*** The delta column increase to be used when outputting the first through to the last matrix column

***LAST\_COLUMN*** The last column of the matrix to be output

***INDEX\_FORMAT\_TYPE*** The format type to be used for the matrix name and indices

**See also:**

[INPUT\\_OUTPUT::MatrixNameIndexFormat](#), [INPUT\\_OUTPUT::MatrixNameIndexFormat](#)

***MATRIX\_NAME\_FORMAT*** The format string to be used to format the matrix name

***ROW\_INDEX\_FORMAT*** The format string to be used to format the row indices

***FIRST\_FORMAT*** The format string to be used for the first line of output

***REPEAT\_FORMAT*** The format type to be used for the second and subsequently repeated lines of output

***ERR*** The error code

***ERROR*** The error string

Definition at line 3922 of file input\_output.f90.

## 7.74 INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE Interface Reference

Write a string, value, string then a value to a given output stream.

### Private Member Functions

- subroutine `WRITE_STRING_TWO_VALUE_C_C` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_C_DP` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_C_INTG` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_C_L` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_C_SP` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_C_VS` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_DP_C` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_DP_DP` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_DP_INTG` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_DP_L` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_DP_SP` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_DP_VS` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_INTG_C` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_INTG_DP` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_INTG_INTG` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_INTG_L` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_INTG_SP` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_INTG_VS` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_L_C` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_L_DP` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_L_INTG` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
- subroutine `WRITE_STRING_TWO_VALUE_L_L` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

- subroutine `WRITE_STRING_TWO_VALUE_L_SP` (*ID*, *FIRST\_STRING*, *FIRST\_VALUE*, *SECOND\_STRING*, *SECOND\_VALUE*, *ERR*, *ERROR,\**)
- subroutine `WRITE_STRING_TWO_VALUE_L_VS` (*ID*, *FIRST\_STRING*, *FIRST\_VALUE*, *SECOND\_STRING*, *SECOND\_VALUE*, *ERR*, *ERROR,\**)
- subroutine `WRITE_STRING_TWO_VALUE_SP_C` (*ID*, *FIRST\_STRING*, *FIRST\_VALUE*, *SECOND\_STRING*, *SECOND\_VALUE*, *ERR*, *ERROR,\**)
- subroutine `WRITE_STRING_TWO_VALUE_SP_DP` (*ID*, *FIRST\_STRING*, *FIRST\_VALUE*, *SECOND\_STRING*, *SECOND\_VALUE*, *ERR*, *ERROR,\**)
- subroutine `WRITE_STRING_TWO_VALUE_SP_INTG` (*ID*, *FIRST\_STRING*, *FIRST\_VALUE*, *SECOND\_STRING*, *SECOND\_VALUE*, *ERR*, *ERROR,\**)
- subroutine `WRITE_STRING_TWO_VALUE_SP_L` (*ID*, *FIRST\_STRING*, *FIRST\_VALUE*, *SECOND\_STRING*, *SECOND\_VALUE*, *ERR*, *ERROR,\**)
- subroutine `WRITE_STRING_TWO_VALUE_SP_SP` (*ID*, *FIRST\_STRING*, *FIRST\_VALUE*, *SECOND\_STRING*, *SECOND\_VALUE*, *ERR*, *ERROR,\**)
- subroutine `WRITE_STRING_TWO_VALUE_SP_VS` (*ID*, *FIRST\_STRING*, *FIRST\_VALUE*, *SECOND\_STRING*, *SECOND\_VALUE*, *ERR*, *ERROR,\**)
- subroutine `WRITE_STRING_TWO_VALUE_VS_C` (*ID*, *FIRST\_STRING*, *FIRST\_VALUE*, *SECOND\_STRING*, *SECOND\_VALUE*, *ERR*, *ERROR,\**)
- subroutine `WRITE_STRING_TWO_VALUE_VS_DP` (*ID*, *FIRST\_STRING*, *FIRST\_VALUE*, *SECOND\_STRING*, *SECOND\_VALUE*, *ERR*, *ERROR,\**)
- subroutine `WRITE_STRING_TWO_VALUE_VS_INTG` (*ID*, *FIRST\_STRING*, *FIRST\_VALUE*, *SECOND\_STRING*, *SECOND\_VALUE*, *ERR*, *ERROR,\**)
- subroutine `WRITE_STRING_TWO_VALUE_VS_L` (*ID*, *FIRST\_STRING*, *FIRST\_VALUE*, *SECOND\_STRING*, *SECOND\_VALUE*, *ERR*, *ERROR,\**)
- subroutine `WRITE_STRING_TWO_VALUE_VS_SP` (*ID*, *FIRST\_STRING*, *FIRST\_VALUE*, *SECOND\_STRING*, *SECOND\_VALUE*, *ERR*, *ERROR,\**)
- subroutine `WRITE_STRING_TWO_VALUE_VS_VS` (*ID*, *FIRST\_STRING*, *FIRST\_VALUE*, *SECOND\_STRING*, *SECOND\_VALUE*, *ERR*, *ERROR,\**)

### 7.74.1 Detailed Description

Write a string, value, string then a value to a given output stream.

Definition at line 88 of file input\_output.f90.

### 7.74.2 Member Function Documentation

**7.74.2.1 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_C\_C (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, CHARACTER(LEN=\*),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, CHARACTER(LEN=\*),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

`BASE_ROUTINES::OutputType`, `BASE_ROUTINES::FileUnits`

*FIRST\_STRING* The first string to be output

*FIRST\_VALUE* The first value to be output

**SECOND\_STRING** The second string to be output  
**SECOND\_VALUE** The second value to be output  
**ERR** The error code  
**ERROR** The error string

Definition at line 480 of file input\_output.f90.

**7.74.2.2 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_C\_DP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, CHARACTER(LEN=\*),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, REAL(DP),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 511 of file input\_output.f90.

**7.74.2.3 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_C\_INTG (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, CHARACTER(LEN=\*),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, INTEGER(INTG),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**FIRST\_VALUE** The first value to be output

**SECOND\_STRING** The second string to be output

**SECOND\_VALUE** The second value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 543 of file input\_output.f90.

---

**7.74.2.4 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_C\_L (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, CHARACTER(LEN=\*),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, LOGICAL,intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 575 of file input\_output.f90.

---

**7.74.2.5 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_C\_SP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, CHARACTER(LEN=\*),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, REAL(SP),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 606 of file input\_output.f90.

---

**7.74.2.6 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_C\_VS (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, CHARACTER(LEN=\*),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, TYPE(VARYING\_STRING),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 638 of file input\_output.f90.

---

**7.74.2.7 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_DP\_C (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, REAL(DP),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, CHARACTER(LEN=\*),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 669 of file input\_output.f90.

---

**7.74.2.8 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_DP\_DP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, REAL(DP),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, REAL(DP),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 701 of file input\_output.f90.

---

**7.74.2.9 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_DP\_INTG (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, REAL(DP),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, INTEGER(INTG),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 737 of file input\_output.f90.

---

**7.74.2.10 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_DP\_L (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, REAL(DP),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, LOGICAL,intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 773 of file input\_output.f90.

---

**7.74.2.11 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_DP\_SP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, REAL(DP),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, REAL(SP),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 808 of file input\_output.f90.

---

**7.74.2.12 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_DP\_VS** (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, REAL(DP),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, TYPE(VARYING\_STRING),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*FIRST\_VALUE* The first value to be output

*SECOND\_STRING* The second string to be output

*SECOND\_VALUE* The second value to be output

*ERR* The error code

*ERROR* The error string

Definition at line 844 of file input\_output.f90.

---

**7.74.2.13 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_INTG\_C** (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, INTEGER(INTG),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, CHARACTER(LEN=\*),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*FIRST\_VALUE* The first value to be output

*SECOND\_STRING* The second string to be output

*SECOND\_VALUE* The second value to be output

*ERR* The error code

*ERROR* The error string

Definition at line 876 of file input\_output.f90.

---

**7.74.2.14 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_INTG\_DP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, INTEGER(INTG),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, REAL(DP),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 908 of file input\_output.f90.

---

**7.74.2.15 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_INTG\_INTG (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, INTEGER(INTG),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, INTEGER(INTG),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 944 of file input\_output.f90.

---

**7.74.2.16 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_INTG\_L (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, INTEGER(INTG),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, LOGICAL,intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 980 of file input\_output.f90.

---

**7.74.2.17 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_INTG\_SP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, INTEGER(INTG),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, REAL(SP),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 1015 of file input\_output.f90.

---

**7.74.2.18 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_INTG\_VS** (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, INTEGER(INTG),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, TYPE(VARYING\_STRING),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*FIRST\_VALUE* The first value to be output

*SECOND\_STRING* The second string to be output

*SECOND\_VALUE* The second value to be output

*ERR* The error code

*ERROR* The error string

Definition at line 1051 of file input\_output.f90.

---

**7.74.2.19 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_L\_C** (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, LOGICAL,intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, CHARACTER(LEN=\*),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*FIRST\_VALUE* The first value to be output

*SECOND\_STRING* The second string to be output

*SECOND\_VALUE* The second value to be output

*ERR* The error code

*ERROR* The error string

Definition at line 1083 of file input\_output.f90.

---

**7.74.2.20 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_L\_DP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, LOGICAL,intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, REAL(DP),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 1115 of file input\_output.f90.

---

**7.74.2.21 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_L\_INTG (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, LOGICAL,intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, INTEGER(INTG),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 1151 of file input\_output.f90.

---

**7.74.2.22 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_L\_L (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, LOGICAL,intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, LOGICAL,intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 1185 of file input\_output.f90.

---

**7.74.2.23 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_L\_SP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, LOGICAL,intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, REAL(SP),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 1220 of file input\_output.f90.

---

**7.74.2.24 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_L\_VS** (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, LOGICAL,intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, TYPE(VARYING\_STRING),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*FIRST\_VALUE* The first value to be output

*SECOND\_STRING* The second string to be output

*SECOND\_VALUE* The second value to be output

*ERR* The error code

*ERROR* The error string

Definition at line 1254 of file input\_output.f90.

---

**7.74.2.25 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_SP\_C** (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, REAL(SP),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, CHARACTER(LEN=\*),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*FIRST\_VALUE* The first value to be output

*SECOND\_STRING* The second string to be output

*SECOND\_VALUE* The second value to be output

*ERR* The error code

*ERROR* The error string

Definition at line 1286 of file input\_output.f90.

---

**7.74.2.26 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_SP\_DP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, REAL(SP),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, REAL(DP),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 1318 of file input\_output.f90.

---

**7.74.2.27 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_SP\_INTG (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, REAL(SP),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, INTEGER(INTG),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 1354 of file input\_output.f90.

---

**7.74.2.28 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_SP\_L (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, REAL(SP),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, LOGICAL,intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 1390 of file input\_output.f90.

---

**7.74.2.29 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_SP\_SP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, REAL(SP),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, REAL(SP),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 1423 of file input\_output.f90.

---

**7.74.2.30 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_SP\_VS** (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, REAL(SP),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, TYPE(VARYING\_STRING),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 1459 of file input\_output.f90.

---

**7.74.2.31 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_VS\_C** (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, TYPE(VARYING\_STRING),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, CHARACTER(LEN=\*),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 1491 of file input\_output.f90.

---

**7.74.2.32 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_VS\_DP** (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, TYPE(VARYING\_STRING),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, REAL(DP),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 1522 of file input\_output.f90.

---

**7.74.2.33 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_VS\_INTG** (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, TYPE(VARYING\_STRING),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, INTEGER(INTG),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 1554 of file input\_output.f90.

---

**7.74.2.34 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_VS\_L** (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, TYPE(VARYING\_STRING),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, LOGICAL,intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 1586 of file input\_output.f90.

---

**7.74.2.35 subroutine INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE::WRITE\_STRING\_TWO\_VALUE\_VS\_SP** (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, TYPE(VARYING\_STRING),intent(in) *FIRST\_VALUE*, CHARACTER(LEN=\*),intent(in) *SECOND\_STRING*, REAL(SP),intent(in) *SECOND\_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

***FIRST\_STRING*** The first string to be output

***FIRST\_VALUE*** The first value to be output

***SECOND\_STRING*** The second string to be output

***SECOND\_VALUE*** The second value to be output

***ERR*** The error code

***ERROR*** The error string

Definition at line 1617 of file input\_output.f90.

```
7.74.2.36 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_-
 STRING_TWO_VALUE_VS_VS (INTEGER(INTG),intent(in) ID,
 CHARACTER(LEN=*),intent(in) FIRST_STRING, TYPE(VARYING_-
 STRING),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
 SECOND_STRING, TYPE(VARYING_STRING),intent(in) SECOND_VALUE,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *) [private]
```

**Parameters:**

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*FIRST\_VALUE* The first value to be output

*SECOND\_STRING* The second string to be output

*SECOND\_VALUE* The second value to be output

*ERR* The error code

*ERROR* The error string

Definition at line 1649 of file input\_output.f90.

## 7.75 INPUT\_OUTPUT::WRITE\_STRING\_VALUE Interface Reference

Write a string followed by a value to a given output stream.

### Private Member Functions

- subroutine [WRITE\\_STRING\\_VALUE\\_C](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)
- subroutine [WRITE\\_STRING\\_VALUE\\_DP](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)
- subroutine [WRITE\\_STRING\\_VALUE\\_INTG](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)
- subroutine [WRITE\\_STRING\\_VALUE\\_LINTG](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)
- subroutine [WRITE\\_STRING\\_VALUE\\_L](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)
- subroutine [WRITE\\_STRING\\_VALUE\\_SP](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)
- subroutine [WRITE\\_STRING\\_VALUE\\_VS](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)

#### 7.75.1 Detailed Description

Write a string followed by a value to a given output stream.

Definition at line 77 of file input\_output.f90.

#### 7.75.2 Member Function Documentation

**7.75.2.1 subroutine INPUT\_OUTPUT::WRITE\_STRING\_VALUE::WRITE\_STRING\_VALUE\_C** (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, CHARACTER(LEN=\*),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

##### Parameters:

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)

*FIRST\_STRING* The first string to be output

*VALUE* The value to be output

*ERR* The error code

*ERROR* The error string

Definition at line 273 of file input\_output.f90.

**7.75.2.2 subroutine INPUT\_OUTPUT::WRITE\_STRING\_VALUE::WRITE\_STRING\_VALUE\_DP** (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*, REAL(DP),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

##### Parameters:

*ID* The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**VALUE** The value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 302 of file input\_output.f90.

**7.75.2.3 subroutine INPUT\_OUTPUT::WRITE\_STRING\_VALUE::WRITE\_STRING\_-  
VALUE\_INTG (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in)  
FIRST\_STRING, INTEGER(INTG),intent(in) *VALUE*, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**VALUE** The value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 332 of file input\_output.f90.

**7.75.2.4 subroutine INPUT\_OUTPUT::WRITE\_STRING\_VALUE::WRITE\_STRING\_-  
VALUE\_L (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in)  
FIRST\_STRING, LOGICAL,intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***ID*** The ID of the output stream. An ID of > 9 specifies file output

**See also:**

[BASE\\_ROUTINES::OutputType](#), [BASE\\_ROUTINES::FileUnits](#)

**FIRST\_STRING** The first string to be output

**VALUE** The value to be output

**ERR** The error code

**ERROR** The error string

Definition at line 392 of file input\_output.f90.

---

**7.75.2.5 subroutine INPUT\_OUTPUT::WRITE\_STRING\_VALUE::WRITE\_STRING\_-  
VALUE\_LINTG (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in)  
*FIRST\_STRING*, INTEGER(LINTG),intent(in) *VALUE*, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:*****ID*** The ID of the output stream. An ID of > 9 specifies file output**See also:**[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)***FIRST\_STRING*** The first string to be output***VALUE*** The value to be output***ERR*** The error code***ERROR*** The error string

Definition at line 362 of file input\_output.f90.

**7.75.2.6 subroutine INPUT\_OUTPUT::WRITE\_STRING\_VALUE::WRITE\_STRING\_-  
VALUE\_SP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in)  
*FIRST\_STRING*, REAL(SP),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:*****ID*** The ID of the output stream. An ID of > 9 specifies file output**See also:**[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)***FIRST\_STRING*** The first string to be output***VALUE*** The value to be output***ERR*** The error code***ERROR*** The error string

Definition at line 421 of file input\_output.f90.

**7.75.2.7 subroutine INPUT\_OUTPUT::WRITE\_STRING\_VALUE::WRITE\_STRING\_VALUE\_-  
VS (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=\*),intent(in) *FIRST\_STRING*,  
TYPE(VARYING\_STRING),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:*****ID*** The ID of the output stream. An ID of > 9 specifies file output**See also:**[BASE\\_ROUTINES::OutputType](#),[BASE\\_ROUTINES::FileUnits](#)***FIRST\_STRING*** The first string to be output***VALUE*** The value to be output***ERR*** The error code***ERROR*** The error string

Definition at line 451 of file input\_output.f90.

## 7.76 INPUT\_OUTPUT::WRITE\_STRING\_VECTOR Interface Reference

Write a string followed by a vector to a specified output stream.

### Private Member Functions

- [WRITE\\_STRING\\_VECTOR\\_DP](#)
- [WRITE\\_STRING\\_VECTOR\\_INTG](#)
- [WRITE\\_STRING\\_VECTOR\\_LINTG](#)
- [WRITE\\_STRING\\_VECTOR\\_L](#)
- [WRITE\\_STRING\\_VECTOR\\_SP](#)

#### 7.76.1 Detailed Description

Write a string followed by a vector to a specified output stream.

Definition at line 179 of file input\_output.f90.

#### 7.76.2 Member Function Documentation

**7.76.2.1 INPUT\_OUTPUT::WRITE\_STRING\_VECTOR::WRITE\_STRING\_VECTOR\_DP ()**  
[private]

**7.76.2.2 INPUT\_OUTPUT::WRITE\_STRING\_VECTOR::WRITE\_STRING\_VECTOR\_INTG ()**  
[private]

**7.76.2.3 INPUT\_OUTPUT::WRITE\_STRING\_VECTOR::WRITE\_STRING\_VECTOR\_L ()**  
[private]

**7.76.2.4 INPUT\_OUTPUT::WRITE\_STRING\_VECTOR::WRITE\_STRING\_VECTOR\_LINTG ()**  
[private]

**7.76.2.5 INPUT\_OUTPUT::WRITE\_STRING\_VECTOR::WRITE\_STRING\_VECTOR\_SP ()**  
[private]

## 7.77 ISO\_VARYING\_STRING::adjustr Interface Reference

### Public Member Functions

- type(varying\_string) **adjustr\_** (string)

#### 7.77.1 Detailed Description

Definition at line 116 of file iso\_varying\_string.f90.

#### 7.77.2 Member Function Documentation

##### 7.77.2.1 type(varying\_string) ISO\_VARYING\_STRING::adjustr::adjustr\_(type(varying\_string),intent(in) string)

Definition at line 875 of file iso\_varying\_string.f90.

## 7.78 ISO\_VARYING\_STRING::assignment(=) Interface Reference

### Public Member Functions

- subroutine `op_assign_CH_VS` (var, exp)
- subroutine `op_assign_VS_CH` (var, exp)
- type(varying\_string) `op_concat_VS_VS` (string\_a, string\_b)
- type(varying\_string) `op_concat_CH_VS` (string\_a, string\_b)
- type(varying\_string) `op_concat_VS_CH` (string\_a, string\_b)
- logical `op_eq_VS_VS` (string\_a, string\_b)
- logical `op_eq_CH_VS` (string\_a, string\_b)
- logical `op_eq_VS_CH` (string\_a, string\_b)
- logical `op_ne_VS_VS` (string\_a, string\_b)
- logical `op_ne_CH_VS` (string\_a, string\_b)
- logical `op_ne_VS_CH` (string\_a, string\_b)
- logical `op_lt_VS_VS` (string\_a, string\_b)
- logical `op_lt_CH_VS` (string\_a, string\_b)
- logical `op_lt_VS_CH` (string\_a, string\_b)
- logical `op_le_VS_VS` (string\_a, string\_b)
- logical `op_le_CH_VS` (string\_a, string\_b)
- logical `op_le_VS_CH` (string\_a, string\_b)
- logical `op_ge_VS_VS` (string\_a, string\_b)
- logical `op_ge_CH_VS` (string\_a, string\_b)
- logical `op_ge_VS_CH` (string\_a, string\_b)
- logical `op_gt_VS_VS` (string\_a, string\_b)
- logical `op_gt_CH_VS` (string\_a, string\_b)
- logical `op_gt_VS_CH` (string\_a, string\_b)
- type(varying\_string) `adjustl_` (string)

#### 7.78.1 Detailed Description

Definition at line 65 of file iso\_varying\_string.f90.

#### 7.78.2 Member Function Documentation

##### 7.78.2.1 type(varying\_string) ISO\_VARYING\_STRING::assignment(=)::adjustl\_(type(varying\_string),intent(in) string)

Definition at line 858 of file iso\_varying\_string.f90.

##### 7.78.2.2 subroutine ISO\_VARYING\_STRING::assignment(=)::op\_assign\_CH\_VS(character(LEN=\*),intent(out) var, type(varying\_string),intent(in) exp)

Definition at line 419 of file iso\_varying\_string.f90.

##### 7.78.2.3 subroutine ISO\_VARYING\_STRING::assignment(=)::op\_assign\_VS\_CH(type(varying\_string),intent(out) var, character(LEN=\*),intent(in) exp)

Definition at line 436 of file iso\_varying\_string.f90.

**7.78.2.4 type(varying\_string) ISO\_VARYING\_STRING::assignment(=)::op\_concat\_CH\_VS  
(character(LEN=\*),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*)**

Definition at line 484 of file iso\_varying\_string.f90.

**7.78.2.5 type(varying\_string) ISO\_VARYING\_STRING::assignment(=)::op\_concat\_VS\_CH  
(type(varying\_string),intent(in) *string\_a*, character(LEN=\*),intent(in) *string\_b*)**

Definition at line 503 of file iso\_varying\_string.f90.

**7.78.2.6 type(varying\_string) ISO\_VARYING\_STRING::assignment(=)::op\_concat\_VS\_VS  
(type(varying\_string),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*)**

Definition at line 453 of file iso\_varying\_string.f90.

**7.78.2.7 logical ISO\_VARYING\_STRING::assignment(=)::op\_eq\_CH\_VS  
(character(LEN=\*),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*)**

Definition at line 540 of file iso\_varying\_string.f90.

**7.78.2.8 logical ISO\_VARYING\_STRING::assignment(=)::op\_eq\_VS\_CH  
(type(varying\_string),intent(in) *string\_a*, character(LEN=\*),intent(in) *string\_b*)**

Definition at line 559 of file iso\_varying\_string.f90.

**7.78.2.9 logical ISO\_VARYING\_STRING::assignment(=)::op\_eq\_VS\_VS  
(type(varying\_string),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*)**

Definition at line 522 of file iso\_varying\_string.f90.

**7.78.2.10 logical ISO\_VARYING\_STRING::assignment(=)::op\_ge\_CH\_VS  
(character(LEN=\*),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*)**

Definition at line 764 of file iso\_varying\_string.f90.

**7.78.2.11 logical ISO\_VARYING\_STRING::assignment(=)::op\_ge\_VS\_CH  
(type(varying\_string),intent(in) *string\_a*, character(LEN=\*),intent(in) *string\_b*)**

Definition at line 783 of file iso\_varying\_string.f90.

**7.78.2.12 logical ISO\_VARYING\_STRING::assignment(=)::op\_ge\_VS\_VS  
(type(varying\_string),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*)**

Definition at line 746 of file iso\_varying\_string.f90.

**7.78.2.13 logical ISO\_VARYING\_STRING::assignment(=)::op\_gt\_CH\_VS  
(character(LEN=\*),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*)**

Definition at line 820 of file iso\_varying\_string.f90.

**7.78.2.14 logical ISO\_VARYING\_STRING::assignment(=)::op\_gt\_VS\_CH  
(type(varying\_string),intent(in) *string\_a*, character(LEN=\*),intent(in) *string\_b*)**

Definition at line 839 of file iso\_varying\_string.f90.

**7.78.2.15 logical ISO\_VARYING\_STRING::assignment(=)::op\_gt\_VS\_VS  
(type(varying\_string),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*)**

Definition at line 802 of file iso\_varying\_string.f90.

**7.78.2.16 logical ISO\_VARYING\_STRING::assignment(=)::op\_le\_CH\_VS  
(character(LEN=\*),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*)**

Definition at line 708 of file iso\_varying\_string.f90.

**7.78.2.17 logical ISO\_VARYING\_STRING::assignment(=)::op\_le\_VS\_CH  
(type(varying\_string),intent(in) *string\_a*, character(LEN=\*),intent(in) *string\_b*)**

Definition at line 727 of file iso\_varying\_string.f90.

**7.78.2.18 logical ISO\_VARYING\_STRING::assignment(=)::op\_le\_VS\_VS  
(type(varying\_string),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*)**

Definition at line 690 of file iso\_varying\_string.f90.

**7.78.2.19 logical ISO\_VARYING\_STRING::assignment(=)::op\_lt\_CH\_VS  
(character(LEN=\*),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*)**

Definition at line 652 of file iso\_varying\_string.f90.

**7.78.2.20 logical ISO\_VARYING\_STRING::assignment(=)::op\_lt\_VS\_CH  
(type(varying\_string),intent(in) *string\_a*, character(LEN=\*),intent(in) *string\_b*)**

Definition at line 671 of file iso\_varying\_string.f90.

**7.78.2.21 logical ISO\_VARYING\_STRING::assignment(=)::op\_lt\_VS\_VS  
(type(varying\_string),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*)**

Definition at line 634 of file iso\_varying\_string.f90.

**7.78.2.22 logical ISO\_VARYING\_STRING::assignment(=)::op\_ne\_CH\_VS**  
**(character(LEN=\*),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*)**

Definition at line 596 of file iso\_varying\_string.f90.

**7.78.2.23 logical ISO\_VARYING\_STRING::assignment(=)::op\_ne\_VS\_CH**  
**(type(varying\_string),intent(in) *string\_a*, character(LEN=\*),intent(in) *string\_b*)**

Definition at line 615 of file iso\_varying\_string.f90.

**7.78.2.24 logical ISO\_VARYING\_STRING::assignment(=)::op\_ne\_VS\_VS**  
**(type(varying\_string),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*)**

Definition at line 578 of file iso\_varying\_string.f90.

## 7.79 ISO\_VARYING\_STRING::char Interface Reference

### Public Member Functions

- function `char_auto` (string)
- character(LEN=length) `char_fixed` (string, length)

#### 7.79.1 Detailed Description

Definition at line 120 of file iso\_varying\_string.f90.

#### 7.79.2 Member Function Documentation

##### 7.79.2.1 function ISO\_VARYING\_STRING::char::char\_auto (type(varying\_string),intent(in) string)

Definition at line 892 of file iso\_varying\_string.f90.

##### 7.79.2.2 character(LEN=length) ISO\_VARYING\_STRING::char::char\_fixed (type(varying\_string),intent(in) string, integer,intent(in) length)

Definition at line 914 of file iso\_varying\_string.f90.

## 7.80 ISO\_VARYING\_STRING::extract Interface Reference

### Public Member Functions

- type(varying\_string) [extract\\_VS](#) (string, start, finish)
- type(varying\_string) [extract\\_CH](#) (string, start, finish)

#### 7.80.1 Detailed Description

Definition at line 218 of file iso\_varying\_string.f90.

#### 7.80.2 Member Function Documentation

##### 7.80.2.1 type(varying\_string) ISO\_VARYING\_STRING::extract::extract\_CH (character(LEN=\*)**,intent(in)** string, integer,intent(in),optional start, integer,intent(in),optional finish)

Definition at line 1901 of file iso\_varying\_string.f90.

##### 7.80.2.2 type(varying\_string) ISO\_VARYING\_STRING::extract::extract\_VS (type(varying\_string),**intent(in)** string, integer,intent(in),optional start, integer,intent(in),optional finish)

Definition at line 1882 of file iso\_varying\_string.f90.

## 7.81 ISO\_VARYING\_STRING::get Interface Reference

### Public Member Functions

- subroutine `get_` (string, maxlen, iostat)
- subroutine `get_unit` (unit, string, maxlen, iostat)
- subroutine `get_set_VS` (string, set, separator, maxlen, iostat)
- subroutine `get_set_CH` (string, set, separator, maxlen, iostat)
- subroutine `get_unit_set_VS` (unit, string, set, separator, maxlen, iostat)
- subroutine `get_unit_set_CH` (unit, string, set, separator, maxlen, iostat)

#### 7.81.1 Detailed Description

Definition at line 195 of file iso\_varying\_string.f90.

#### 7.81.2 Member Function Documentation

**7.81.2.1 subroutine ISO\_VARYING\_STRING::get::get\_ (type(varying\_string),intent(out) *string*, integer,intent(in),optional *maxlen*, integer,intent(out),optional *iostat*)**

Definition at line 1455 of file iso\_varying\_string.f90.

**7.81.2.2 subroutine ISO\_VARYING\_STRING::get::get\_set\_CH (type(varying\_string),intent(out) *string*, character(LEN=\*)intent(in) *set*, type(varying\_string),intent(out),optional *separator*, integer,intent(in),optional *maxlen*, integer,intent(out),optional *iostat*)**

Definition at line 1583 of file iso\_varying\_string.f90.

**7.81.2.3 subroutine ISO\_VARYING\_STRING::get::get\_set\_VS (type(varying\_string),intent(out) *string*, type(varying\_string),intent(in) *set*, type(varying\_string),intent(out),optional *separator*, integer,intent(in),optional *maxlen*, integer,intent(out),optional *iostat*)**

Definition at line 1562 of file iso\_varying\_string.f90.

**7.81.2.4 subroutine ISO\_VARYING\_STRING::get::get\_unit (integer,intent(in) *unit*, type(varying\_string),intent(out) *string*, integer,intent(in),optional *maxlen*, integer,intent(out),optional *iostat*)**

Definition at line 1508 of file iso\_varying\_string.f90.

**7.81.2.5 subroutine ISO\_VARYING\_STRING::get::get\_unit\_set\_CH (integer,intent(in) *unit*, type(varying\_string),intent(out) *string*, character(LEN=\*)intent(in) *set*, type(varying\_string),intent(out),optional *separator*, integer,intent(in),optional *maxlen*, integer,intent(out),optional *iostat*)**

Definition at line 1663 of file iso\_varying\_string.f90.

**7.81.2.6 subroutine ISO\_VARYING\_STRING::get::get\_unit\_set\_VS** (**integer,intent(in)**  
***unit*, type(varying\_string),intent(out) *string*, type(varying\_string),intent(in) *set*,**  
**type(varying\_string),intent(out),optional *separator*, integer,intent(in),optional *maxlen*,**  
**integer,intent(out),optional *iostat***)

Definition at line 1641 of file iso\_varying\_string.f90.

## 7.82 ISO\_VARYING\_STRING::iachar Interface Reference

### Public Member Functions

- `integer iachar_(c)`

#### 7.82.1 Detailed Description

Definition at line 125 of file iso\_varying\_string.f90.

#### 7.82.2 Member Function Documentation

##### 7.82.2.1 `integer ISO_VARYING_STRING::iachar::iachar_(type(varying_string),intent(in) c)`

Definition at line 933 of file iso\_varying\_string.f90.

## 7.83 ISO\_VARYING\_STRING::ichar Interface Reference

### Public Member Functions

- `integer ichar_(c)`

#### 7.83.1 Detailed Description

Definition at line 129 of file iso\_varying\_string.f90.

#### 7.83.2 Member Function Documentation

##### 7.83.2.1 `integer ISO_VARYING_STRING::ichar::ichar_(type(varying_string),intent(in) c)`

Definition at line 951 of file iso\_varying\_string.f90.

## 7.84 ISO\_VARYING\_STRING::index Interface Reference

### Public Member Functions

- `integer index_VS_VS` (`string, substring, back`)
- `integer index_CH_VS` (`string, substring, back`)
- `integer index_VS_CH` (`string, substring, back`)

#### 7.84.1 Detailed Description

Definition at line 133 of file iso\_varying\_string.f90.

#### 7.84.2 Member Function Documentation

##### 7.84.2.1 `integer ISO_VARYING_STRING::index::index_CH_VS` (`character(LEN=*)`,`intent(in)` `string, type(varying_string),intent(in) substring, logical,intent(in),optional back`)

Definition at line 989 of file iso\_varying\_string.f90.

##### 7.84.2.2 `integer ISO_VARYING_STRING::index::index_VS_CH` (`type(varying_string)`,`intent(in)` `string, character(LEN=*)`,`intent(in) substring, logical,intent(in),optional back`)

Definition at line 1009 of file iso\_varying\_string.f90.

##### 7.84.2.3 `integer ISO_VARYING_STRING::index::index_VS_VS` (`type(varying_string)`,`intent(in)` `string, type(varying_string),intent(in) substring, logical,intent(in),optional back`)

Definition at line 969 of file iso\_varying\_string.f90.

## 7.85 ISO\_VARYING\_STRING::insert Interface Reference

### Public Member Functions

- type(varying\_string) [insert\\_VS\\_VS](#) (string, start, substring)
- type(varying\_string) [insert\\_CH\\_VS](#) (string, start, substring)
- type(varying\_string) [insert\\_VS\\_CH](#) (string, start, substring)
- type(varying\_string) [insert\\_CH\\_CH](#) (string, start, substring)

#### 7.85.1 Detailed Description

Definition at line 223 of file iso\_varying\_string.f90.

#### 7.85.2 Member Function Documentation

##### 7.85.2.1 type(varying\_string) ISO\_VARYING\_STRING::insert::insert\_CH\_CH (character(LEN=\*),intent(in) string, integer,intent(in) start, character(LEN=\*),intent(in) substring)

Definition at line 1992 of file iso\_varying\_string.f90.

##### 7.85.2.2 type(varying\_string) ISO\_VARYING\_STRING::insert::insert\_CH\_VS (character(LEN=\*),intent(in) string, integer,intent(in) start, type(varying\_string),intent(in) substring)

Definition at line 1954 of file iso\_varying\_string.f90.

##### 7.85.2.3 type(varying\_string) ISO\_VARYING\_STRING::insert::insert\_VS\_CH (type(varying\_- string),intent(in) string, integer,intent(in) start, character(LEN=\*),intent(in) substring)

Definition at line 1973 of file iso\_varying\_string.f90.

##### 7.85.2.4 type(varying\_string) ISO\_VARYING\_STRING::insert::insert\_VS\_VS (type(varying\_- string),intent(in) string, integer,intent(in) start, type(varying\_string),intent(in) substring)

Definition at line 1935 of file iso\_varying\_string.f90.

## 7.86 ISO\_VARYING\_STRING::len Interface Reference

### Public Member Functions

- `integer len_(string)`

#### 7.86.1 Detailed Description

Definition at line 139 of file iso\_varying\_string.f90.

#### 7.86.2 Member Function Documentation

##### 7.86.2.1 `integer ISO_VARYING_STRING::len::len_ (type(varying_string),intent(in) string)`

Definition at line 398 of file iso\_varying\_string.f90.

## 7.87 ISO\_VARYING\_STRING::len\_trim Interface Reference

### Public Member Functions

- `integer len_trim_(string)`

#### 7.87.1 Detailed Description

Definition at line 143 of file iso\_varying\_string.f90.

#### 7.87.2 Member Function Documentation

##### 7.87.2.1 `integer ISO_VARYING_STRING::len_trim::len_trim_ (type(varying_string),intent(in) string)`

Definition at line 1029 of file iso\_varying\_string.f90.

## 7.88 ISO\_VARYING\_STRING::lge Interface Reference

### Public Member Functions

- [logical lge\\_VS\\_VS \(string\\_a, string\\_b\)](#)
- [logical lge\\_CH\\_VS \(string\\_a, string\\_b\)](#)
- [logical lge\\_VS\\_CH \(string\\_a, string\\_b\)](#)

#### 7.88.1 Detailed Description

Definition at line 147 of file iso\_varying\_string.f90.

#### 7.88.2 Member Function Documentation

##### 7.88.2.1 logical ISO\_VARYING\_STRING::lge::lge\_CH\_VS (character(LEN=\*)**,intent(in)** *string\_a*, type(varying\_string)**,intent(in)** *string\_b*)

Definition at line 1068 of file iso\_varying\_string.f90.

##### 7.88.2.2 logical ISO\_VARYING\_STRING::lge::lge\_VS\_CH (type(varying\_string)**,intent(in)** *string\_a*, character(LEN=\*)**,intent(in)** *string\_b*)

Definition at line 1087 of file iso\_varying\_string.f90.

##### 7.88.2.3 logical ISO\_VARYING\_STRING::lge::lge\_VS\_VS (type(varying\_string)**,intent(in)** *string\_a*, type(varying\_string)**,intent(in)** *string\_b*)

Definition at line 1050 of file iso\_varying\_string.f90.

## 7.89 ISO\_VARYING\_STRING::lgt Interface Reference

### Public Member Functions

- [logical lgt\\_VS\\_VS](#) (string\_a, string\_b)
- [logical lgt\\_CH\\_VS](#) (string\_a, string\_b)
- [logical lgt\\_VS\\_CH](#) (string\_a, string\_b)

#### 7.89.1 Detailed Description

Definition at line 153 of file iso\_varying\_string.f90.

#### 7.89.2 Member Function Documentation

##### 7.89.2.1 logical ISO\_VARYING\_STRING::lgt::lgt\_CH\_VS (character(LEN=\*),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*)

Definition at line 1124 of file iso\_varying\_string.f90.

##### 7.89.2.2 logical ISO\_VARYING\_STRING::lgt::lgt\_VS\_CH (type(varying\_string),intent(in) *string\_a*, character(LEN=\*),intent(in) *string\_b*)

Definition at line 1143 of file iso\_varying\_string.f90.

##### 7.89.2.3 logical ISO\_VARYING\_STRING::lgt::lgt\_VS\_VS (type(varying\_string),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*)

Definition at line 1106 of file iso\_varying\_string.f90.

## 7.90 ISO\_VARYING\_STRING::lle Interface Reference

### Public Member Functions

- [logical lle\\_VS\\_VS](#) (string\_a, string\_b)
- [logical lle\\_CH\\_VS](#) (string\_a, string\_b)
- [logical lle\\_VS\\_CH](#) (string\_a, string\_b)

#### 7.90.1 Detailed Description

Definition at line 159 of file iso\_varying\_string.f90.

#### 7.90.2 Member Function Documentation

##### 7.90.2.1 logical ISO\_VARYING\_STRING::lle::lle\_CH\_VS (character(LEN=\*),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*)

Definition at line 1180 of file iso\_varying\_string.f90.

##### 7.90.2.2 logical ISO\_VARYING\_STRING::lle::lle\_VS\_CH (type(varying\_string),intent(in) *string\_a*, character(LEN=\*),intent(in) *string\_b*)

Definition at line 1199 of file iso\_varying\_string.f90.

##### 7.90.2.3 logical ISO\_VARYING\_STRING::lle::lle\_VS\_VS (type(varying\_string),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*)

Definition at line 1162 of file iso\_varying\_string.f90.

## 7.91 ISO\_VARYING\_STRING::llt Interface Reference

### Public Member Functions

- [logical llt\\_VS\\_VS](#) (string\_a, string\_b)
- [logical llt\\_CH\\_VS](#) (string\_a, string\_b)
- [logical llt\\_VS\\_CH](#) (string\_a, string\_b)

#### 7.91.1 Detailed Description

Definition at line 165 of file iso\_varying\_string.f90.

#### 7.91.2 Member Function Documentation

##### 7.91.2.1 logical ISO\_VARYING\_STRING::llt::llt\_CH\_VS (character(LEN=\*),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*)

Definition at line 1236 of file iso\_varying\_string.f90.

##### 7.91.2.2 logical ISO\_VARYING\_STRING::llt::llt\_VS\_CH (type(varying\_string),intent(in) *string\_a*, character(LEN=\*),intent(in) *string\_b*)

Definition at line 1255 of file iso\_varying\_string.f90.

##### 7.91.2.3 logical ISO\_VARYING\_STRING::llt::llt\_VS\_VS (type(varying\_string),intent(in) *string\_a*, type(varying\_string),intent(in) *string\_b*)

Definition at line 1218 of file iso\_varying\_string.f90.

## 7.92 ISO\_VARYING\_STRING::put Interface Reference

### Public Member Functions

- subroutine [put\\_VS](#) (string, iostat)
- subroutine [put\\_CH](#) (string, iostat)
- subroutine [put\\_unit\\_VS](#) (unit, string, iostat)
- subroutine [put\\_unit\\_CH](#) (unit, string, iostat)

#### 7.92.1 Detailed Description

Definition at line 204 of file iso\_varying\_string.f90.

#### 7.92.2 Member Function Documentation

##### 7.92.2.1 subroutine ISO\_VARYING\_STRING::put::put\_CH (character(LEN=\*),intent(in) *string*, integer,intent(out),optional *iostat*)

Definition at line 1738 of file iso\_varying\_string.f90.

##### 7.92.2.2 subroutine ISO\_VARYING\_STRING::put::put\_unit\_CH (integer,intent(in) *unit*, character(LEN=\*),intent(in) *string*, integer,intent(out),optional *iostat*)

Definition at line 1777 of file iso\_varying\_string.f90.

##### 7.92.2.3 subroutine ISO\_VARYING\_STRING::put::put\_unit\_VS (integer,intent(in) *unit*, type(varying\_string),intent(in) *string*, integer,intent(out),optional *iostat*)

Definition at line 1758 of file iso\_varying\_string.f90.

##### 7.92.2.4 subroutine ISO\_VARYING\_STRING::put::put\_VS (type(varying\_string),intent(in) *string*, integer,intent(out),optional *iostat*)

Definition at line 1722 of file iso\_varying\_string.f90.

## 7.93 ISO\_VARYING\_STRING::put\_line Interface Reference

### Public Member Functions

- subroutine [put\\_line\\_VS](#) (string, iostat)
- subroutine [put\\_line\\_CH](#) (string, iostat)
- subroutine [put\\_line\\_unit\\_VS](#) (unit, string, iostat)
- subroutine [put\\_line\\_unit\\_CH](#) (unit, string, iostat)

#### 7.93.1 Detailed Description

Definition at line 211 of file iso\_varying\_string.f90.

#### 7.93.2 Member Function Documentation

##### 7.93.2.1 subroutine ISO\_VARYING\_STRING::put\_line::put\_line\_CH (character(LEN=\*),intent(in) *string*, integer,intent(out),optional *iostat*)

Definition at line 1818 of file iso\_varying\_string.f90.

##### 7.93.2.2 subroutine ISO\_VARYING\_STRING::put\_line::put\_line\_unit\_CH (integer,intent(in) *unit*, character(LEN=\*),intent(in) *string*, integer,intent(out),optional *iostat*)

Definition at line 1859 of file iso\_varying\_string.f90.

##### 7.93.2.3 subroutine ISO\_VARYING\_STRING::put\_line::put\_line\_unit\_VS (integer,intent(in) *unit*, type(varying\_string),intent(in) *string*, integer,intent(out),optional *iostat*)

Definition at line 1840 of file iso\_varying\_string.f90.

##### 7.93.2.4 subroutine ISO\_VARYING\_STRING::put\_line::put\_line\_VS (type(varying\_string),intent(in) *string*, integer,intent(out),optional *iostat*)

Definition at line 1800 of file iso\_varying\_string.f90.

## 7.94 ISO\_VARYING\_STRING::remove Interface Reference

### Public Member Functions

- TYPE(varying\_string) [remove\\_VS](#) (string, start, finish)
- type(varying\_string) [remove\\_CH](#) (string, start, finish)

#### 7.94.1 Detailed Description

Definition at line 230 of file iso\_varying\_string.f90.

#### 7.94.2 Member Function Documentation

##### 7.94.2.1 [type\(varying\\_string\) ISO\\_VARYING\\_STRING::remove::remove\\_CH](#) (character(LEN=\*)**,intent(in)** *string*, **integer,intent(in),optional** *start*, **integer,intent(in),optional** *finish*)

Definition at line 2035 of file iso\_varying\_string.f90.

##### 7.94.2.2 [TYPE\(varying\\_string\) ISO\\_VARYING\\_STRING::remove::remove\\_VS](#) (**type(varying\_string),intent(in)** *string*, **integer,intent(in),optional** *start*, **integer,intent(in),optional** *finish*)

Definition at line 2016 of file iso\_varying\_string.f90.

## 7.95 ISO\_VARYING\_STRING::repeat Interface Reference

### Public Member Functions

- type(varying\_string) [repeat\\_](#) (string, ncopies)

#### 7.95.1 Detailed Description

Definition at line 171 of file iso\_varying\_string.f90.

#### 7.95.2 Member Function Documentation

##### 7.95.2.1 type(varying\_string) ISO\_VARYING\_STRING::repeat::repeat\_(type(varying\_string),intent(in) string, integer,intent(in) ncopies)

Definition at line 1274 of file iso\_varying\_string.f90.

## 7.96 ISO\_VARYING\_STRING::replace Interface Reference

### Public Member Functions

- type(varying\_string) `replace_VS_VS_auto` (string, start, substring)
- type(varying\_string) `replace_CH_VS_auto` (string, start, substring)
- type(varying\_string) `replace_VS_CH_auto` (string, start, substring)
- type(varying\_string) `replace_CH_CH_auto` (string, start, substring)
- type(varying\_string) `replace_VS_VS_fixed` (string, start, finish, substring)
- type(varying\_string) `replace_CH_VS_fixed` (string, start, finish, substring)
- type(varying\_string) `replace_VS_CH_fixed` (string, start, finish, substring)
- type(varying\_string) `replace_CH_CH_fixed` (string, start, finish, substring)
- type(varying\_string) `replace_VS_VS_VS_target` (string, target, substring, every, back)
- type(varying\_string) `replace_CH_VS_VS_target` (string, target, substring, every, back)
- type(varying\_string) `replace_VS_CH_VS_target` (string, target, substring, every, back)
- type(varying\_string) `replace_CH_CH_VS_target` (string, target, substring, every, back)
- type(varying\_string) `replace_VS_VS_CH_target` (string, target, substring, every, back)
- type(varying\_string) `replace_CH_VS_CH_target` (string, target, substring, every, back)
- type(varying\_string) `replace_VS_CH_CH_target` (string, target, substring, every, back)
- type(varying\_string) `replace_CH_CH_CH_target` (string, target, substring, every, back)

### 7.96.1 Detailed Description

Definition at line 235 of file iso\_varying\_string.f90.

### 7.96.2 Member Function Documentation

**7.96.2.1 type(varying\_string) ISO\_VARYING\_STRING::replace::replace\_CH\_CH\_auto**  
`(character(LEN=*)intent(in) string, integer,intent(in) start, character(LEN=*)intent(in) substring)`

Definition at line 2133 of file iso\_varying\_string.f90.

**7.96.2.2 type(varying\_string) ISO\_VARYING\_STRING::replace::replace\_CH\_CH\_CH\_target**  
`(character(LEN=*)intent(in) string, character(LEN=*)intent(in) target,`  
`character(LEN=*)intent(in) substring, logical,intent(in),optional every,`  
`logical,intent(in),optional back)`

Definition at line 2410 of file iso\_varying\_string.f90.

**7.96.2.3 type(varying\_string) ISO\_VARYING\_STRING::replace::replace\_CH\_CH\_fixed**  
`(character(LEN=*)intent(in) string, integer,intent(in) start, integer,intent(in) finish,`  
`character(LEN=*)intent(in) substring)`

Definition at line 2218 of file iso\_varying\_string.f90.

---

**7.96.2.4 type(varying\_string) ISO\_VARYING\_STRING::replace::replace\_CH\_CH\_VS\_target**  
 (character(LEN=\*),intent(in) *string*, character(LEN=\*),intent(in) *target*,  
 type(varying\_string),intent(in) *substring*, logical,intent(in),optional *every*,  
 logical,intent(in),optional *back*)

Definition at line 2318 of file iso\_varying\_string.f90.

**7.96.2.5 type(varying\_string) ISO\_VARYING\_STRING::replace::replace\_CH\_-**  
**VS\_auto** (character(LEN=\*),intent(in) *string*, integer,intent(in) *start*,  
 type(varying\_string),intent(in) *substring*)

Definition at line 2093 of file iso\_varying\_string.f90.

**7.96.2.6 type(varying\_string) ISO\_VARYING\_STRING::replace::replace\_CH\_VS\_CH\_target**  
 (character(LEN=\*),intent(in) *string*, type(varying\_string),intent(in) *target*,  
 character(LEN=\*),intent(in) *substring*, logical,intent(in),optional *every*,  
 logical,intent(in),optional *back*)

Definition at line 2364 of file iso\_varying\_string.f90.

**7.96.2.7 type(varying\_string) ISO\_VARYING\_STRING::replace::replace\_CH\_VS\_fixed**  
 (character(LEN=\*),intent(in) *string*, integer,intent(in) *start*, integer,intent(in) *finish*,  
 type(varying\_string),intent(in) *substring*)

Definition at line 2176 of file iso\_varying\_string.f90.

**7.96.2.8 type(varying\_string) ISO\_VARYING\_STRING::replace::replace\_CH\_VS\_VS\_target**  
 (character(LEN=\*),intent(in) *string*, type(varying\_string),intent(in) *target*,  
 type(varying\_string),intent(in) *substring*, logical,intent(in),optional *every*,  
 logical,intent(in),optional *back*)

Definition at line 2272 of file iso\_varying\_string.f90.

**7.96.2.9 type(varying\_string) ISO\_VARYING\_STRING::replace::replace\_VS\_-**  
**CH\_auto** (type(varying\_string),intent(in) *string*, integer,intent(in) *start*,  
 character(LEN=\*),intent(in) *substring*)

Definition at line 2113 of file iso\_varying\_string.f90.

**7.96.2.10 type(varying\_string) ISO\_VARYING\_STRING::replace::replace\_VS\_CH\_CH\_target**  
 (type(varying\_string),intent(in) *string*, character(LEN=\*),intent(in) *target*,  
 character(LEN=\*),intent(in) *substring*, logical,intent(in),optional *every*,  
 logical,intent(in),optional *back*)

Definition at line 2387 of file iso\_varying\_string.f90.

**7.96.2.11 type(varying\_string) ISO\_VARYING\_STRING::replace::replace\_VS\_CH\_fixed  
(type(varying\_string),intent(in) *string*, integer,intent(in) *start*, integer,intent(in) *finish*,  
character(LEN=\*)intent(in) *substring*)**

Definition at line 2197 of file iso\_varying\_string.f90.

**7.96.2.12 type(varying\_string) ISO\_VARYING\_STRING::replace::replace\_VS\_CH\_VS\_target  
(type(varying\_string),intent(in) *string*, character(LEN=\*)intent(in) *target*,  
type(varying\_string),intent(in) *substring*, logical,intent(in),optional *every*,  
logical,intent(in),optional *back*)**

Definition at line 2295 of file iso\_varying\_string.f90.

**7.96.2.13 type(varying\_string) ISO\_VARYING\_STRING::replace::replace\_VS\_VS\_auto  
(type(varying\_string),intent(in) *string*, integer,intent(in) *start*,  
type(varying\_string),intent(in) *substring*)**

Definition at line 2073 of file iso\_varying\_string.f90.

**7.96.2.14 type(varying\_string) ISO\_VARYING\_STRING::replace::replace\_VS\_VS\_CH\_target  
(type(varying\_string),intent(in) *string*, type(varying\_string),intent(in) *target*,  
character(LEN=\*)intent(in) *substring*, logical,intent(in),optional *every*,  
logical,intent(in),optional *back*)**

Definition at line 2341 of file iso\_varying\_string.f90.

**7.96.2.15 type(varying\_string) ISO\_VARYING\_STRING::replace::replace\_VS\_VS\_fixed  
(type(varying\_string),intent(in) *string*, integer,intent(in) *start*, integer,intent(in) *finish*,  
type(varying\_string),intent(in) *substring*)**

Definition at line 2153 of file iso\_varying\_string.f90.

**7.96.2.16 type(varying\_string) ISO\_VARYING\_STRING::replace::replace\_VS\_VS\_VS\_target  
(type(varying\_string),intent(in) *string*, type(varying\_string),intent(in) *target*,  
type(varying\_string),intent(in) *substring*, logical,intent(in),optional *every*,  
logical,intent(in),optional *back*)**

Definition at line 2249 of file iso\_varying\_string.f90.

## 7.97 ISO\_VARYING\_STRING::scan Interface Reference

### Public Member Functions

- [integer scan\\_VS\\_VS](#) (string, set, back)
- [integer scan\\_CH\\_VS](#) (string, set, back)
- [integer scan\\_VS\\_CH](#) (string, set, back)

#### 7.97.1 Detailed Description

Definition at line 175 of file iso\_varying\_string.f90.

#### 7.97.2 Member Function Documentation

##### 7.97.2.1 [integer ISO\\_VARYING\\_STRING::scan::scan\\_CH\\_VS](#) (`character(LEN=*)`,`intent(in)` string, `type(varying_string)`,`intent(in)` set, logical,`intent(in)`,`optional` back)

Definition at line 1312 of file iso\_varying\_string.f90.

##### 7.97.2.2 [integer ISO\\_VARYING\\_STRING::scan::scan\\_VS\\_CH](#) (`type(varying_string)`,`intent(in)` string, `character(LEN=*)`,`intent(in)` set, logical,`intent(in)`,`optional` back)

Definition at line 1332 of file iso\_varying\_string.f90.

##### 7.97.2.3 [integer ISO\\_VARYING\\_STRING::scan::scan\\_VS\\_VS](#) (`type(varying_string)`,`intent(in)` string, `type(varying_string)`,`intent(in)` set, logical,`intent(in)`,`optional` back)

Definition at line 1292 of file iso\_varying\_string.f90.

## 7.98 ISO\_VARYING\_STRING::split Interface Reference

### Public Member Functions

- subroutine [split\\_VS](#) (string, word, set, separator, back)
- subroutine [split\\_CH](#) (string, word, set, separator, back)

#### 7.98.1 Detailed Description

Definition at line 254 of file iso\_varying\_string.f90.

#### 7.98.2 Member Function Documentation

**7.98.2.1 subroutine ISO\_VARYING\_STRING::split::split\_CH** (*type(varying\_string),intent(inout) string*, *type(varying\_string),intent(out) word*, *character(LEN=\*)intent(in) set*,  
*type(varying\_string),intent(out),optional separator*, *logical,intent(in),optional back*)

Definition at line 2514 of file iso\_varying\_string.f90.

**7.98.2.2 subroutine ISO\_VARYING\_STRING::split::split\_VS** (*type(varying\_string),intent(inout) string*, *type(varying\_string),intent(out) word*, *type(varying\_string),intent(in) set*,  
*type(varying\_string),intent(out),optional separator*, *logical,intent(in),optional back*)

Definition at line 2494 of file iso\_varying\_string.f90.

## 7.99 ISO\_VARYING\_STRING::trim Interface Reference

### Public Member Functions

- type(varying\_string) **trim\_** (string)

#### 7.99.1 Detailed Description

Definition at line 181 of file iso\_varying\_string.f90.

#### 7.99.2 Member Function Documentation

##### 7.99.2.1 type(varying\_string) ISO\_VARYING\_STRING::trim::trim\_ (type(varying\_string),intent(in) string)

Definition at line 1352 of file iso\_varying\_string.f90.

## 7.100 ISO\_VARYING\_STRING::var\_str Interface Reference

### Public Member Functions

- type(varying\_string) [var\\_str\\_ \(char\)](#)

#### 7.100.1 Detailed Description

Definition at line 191 of file iso\_varying\_string.f90.

#### 7.100.2 Member Function Documentation

##### 7.100.2.1 [type\(varying\\_string\) ISO\\_VARYING\\_STRING::var\\_str::var\\_str\\_\(character\(LEN=\\*\),intent\(in\) char\)](#)

Definition at line 1429 of file iso\_varying\_string.f90.

## 7.101 ISO\_VARYING\_STRING::VARYING\_STRING Struct Reference

### Public Attributes

- character(LEN=1), dimension(:), allocatable [chars](#)

#### 7.101.1 Detailed Description

Definition at line 58 of file iso\_varying\_string.f90.

#### 7.101.2 Member Data Documentation

##### 7.101.2.1 character(LEN=1),dimension(:),allocatable ISO\_VARYING\_STRING::VARYING\_STRING::chars

Definition at line 60 of file iso\_varying\_string.f90.

## 7.102 ISO\_VARYING\_STRING::verify Interface Reference

### Public Member Functions

- `integer verify_VS_VS (string, set, back)`
- `integer verify_CH_VS (string, set, back)`
- `integer verify_VS_CH (string, set, back)`

#### 7.102.1 Detailed Description

Definition at line 185 of file iso\_varying\_string.f90.

#### 7.102.2 Member Function Documentation

##### 7.102.2.1 `integer ISO_VARYING_STRING::verify::verify_CH_VS (character(LEN=*)intent(in) string, type(varying_string),intent(in) set, logical,intent(in),optional back)`

Definition at line 1389 of file iso\_varying\_string.f90.

##### 7.102.2.2 `integer ISO_VARYING_STRING::verify::verify_VS_CH (type(varying_string),intent(in) string, character(LEN=*)intent(in) set, logical,intent(in),optional back)`

Definition at line 1409 of file iso\_varying\_string.f90.

##### 7.102.2.3 `integer ISO_VARYING_STRING::verify::verify_VS_VS (type(varying_string),intent(in) string, type(varying_string),intent(in) set, logical,intent(in),optional back)`

Definition at line 1369 of file iso\_varying\_string.f90.

## 7.103 LAPACK::interface Interface Reference

### Public Member Functions

- subroutine [DGESV](#) (N, NRHS, A, LDA, IPIV, B, LDB, INFO)

#### 7.103.1 Detailed Description

Definition at line 50 of file lapack.f90.

#### 7.103.2 Member Function Documentation

- 7.103.2.1 subroutine LAPACK::interface::DGESV (INTEGER(INTG),intent(in) *N*,  
INTEGER(INTG),intent(in) *NRHS*, REAL(DP),dimension(lda,\*),intent(inout) *A*,  
INTEGER(INTG),intent(in) *LDA*, INTEGER(INTG),dimension(\*),intent(out) *IPIV*,  
REAL(DP),dimension(lbd,\*),intent(inout) *B*, INTEGER(INTG),intent(in) *LDB*,  
INTEGER(INTG),intent(out) *INFO*)

Definition at line 52 of file lapack.f90.

## 7.104 LISTS::LIST\_DETACH\_AND\_DESTROY Interface Reference

Detaches the list values from a list and returns them as a pointer to a array of base type before destroying the list.

### Private Member Functions

- subroutine [LIST\\_DETACH\\_AND\\_DESTROY\\_INTG](#) (LIST, NUMBER\_IN\_LIST, LIST\_VALUES, ERR, ERROR,\*)
- subroutine [LIST\\_DETACH\\_AND\\_DESTROY\\_SP](#) (LIST, NUMBER\_IN\_LIST, LIST\_VALUES, ERR, ERROR,\*)
- subroutine [LIST\\_DETACH\\_AND\\_DESTROY\\_DP](#) (LIST, NUMBER\_IN\_LIST, LIST\_VALUES, ERR, ERROR,\*)

#### 7.104.1 Detailed Description

Detaches the list values from a list and returns them as a pointer to a array of base type before destroying the list.

##### See also:

[LISTS](#).

Definition at line 126 of file lists.f90.

#### 7.104.2 Member Function Documentation

**7.104.2.1 subroutine LISTS::LIST\_DETACH\_AND\_DESTROY::LIST\_DETACH\_AND\_DESTROY\_DP** (TYPE(List\_Type),pointer *LIST*, INTEGER(INTG),intent(out) *NUMBER\_IN\_LIST*, REAL(DP),dimension(:),pointer *LIST\_VALUES*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

##### Parameters:

***LIST*** The pointer to the list

***NUMBER\_IN\_LIST*** On exit, the number in the list that has been detached.

***LIST\_VALUES*** On exit, a pointer to the detached list. Must not be associated on entry.

***ERR*** The error code

***ERROR*** The error string

Definition at line 931 of file lists.f90.

**7.104.2.2 subroutine LISTS::LIST\_DETACH\_AND\_DESTROY::LIST\_DETACH\_AND\_-  
DESTROY\_INTG (TYPE(LIST\_TYPE),pointer LIST, INTEGER(INTG),intent(out)  
NUMBER\_IN\_LIST, INTEGER(INTG),dimension(:),pointer LIST\_VALUES,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*) [private]**

**Parameters:**

**LIST** The pointer to the list

**NUMBER\_IN\_LIST** On exit, the number in the list that has been detached.

**LIST\_VALUES** On exit, a pointer to the detached list. Must not be associated on entry.

**ERR** The error code

**ERROR** The error string

Definition at line 830 of file lists.f90.

**7.104.2.3 subroutine LISTS::LIST\_DETACH\_AND\_DESTROY::LIST\_DETACH\_AND\_-  
DESTROY\_SP (TYPE(LIST\_TYPE),pointer LIST, INTEGER(INTG),intent(out)  
NUMBER\_IN\_LIST, REAL(SP),dimension(:),pointer LIST\_VALUES,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*) [private]**

**Parameters:**

**LIST** The pointer to the list

**NUMBER\_IN\_LIST** On exit, the number in the list that has been detached.

**LIST\_VALUES** On exit, a pointer to the detached list. Must not be associated on entry.

**ERR** The error code

**ERROR** The error string

Definition at line 881 of file lists.f90.

## 7.105 LISTS::LIST\_ITEM\_ADD Interface Reference

Adds an item to the end of a list.

### Private Member Functions

- subroutine [LIST\\_ITEM\\_ADD\\_INTG1](#) (LIST, ITEM, ERR, ERROR,\*)
- subroutine [LIST\\_ITEM\\_ADD\\_SP1](#) (LIST, ITEM, ERR, ERROR,\*)
- subroutine [LIST\\_ITEM\\_ADD\\_DP1](#) (LIST, ITEM, ERR, ERROR,\*)

### 7.105.1 Detailed Description

Adds an item to the end of a list.

#### See also:

[LISTS](#).

Definition at line 112 of file lists.f90.

### 7.105.2 Member Function Documentation

**7.105.2.1 subroutine LISTS::LIST\_ITEM\_ADD::LIST\_ITEM\_ADD\_DP1 (TYPE(LIST\_TYPE),pointer LIST, REAL(DP),intent(in) ITEM, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

#### Parameters:

*LIST* A pointer to the list

*ITEM* The item to add

*ERR* The error code

*ERROR* The error string

Definition at line 566 of file lists.f90.

**7.105.2.2 subroutine LISTS::LIST\_ITEM\_ADD::LIST\_ITEM\_ADD\_INTG1 (TYPE(LIST\_TYPE),pointer LIST, INTEGER(INTG),intent(in) ITEM, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

#### Parameters:

*LIST* A pointer to the list

*ITEM* The item to add

*ERR* The error code

*ERROR* The error string

Definition at line 457 of file lists.f90.

**7.105.2.3 subroutine LISTS::LIST\_ITEM\_ADD::LIST\_ITEM\_ADD\_SP1 (TYPE(LIST\_-  
TYPE),pointer *LIST*, REAL(SP),intent(in) *ITEM*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

*LIST* A pointer to the list

*ITEM* The item to add

*ERR* The error code

*ERROR* The error string

Definition at line 511 of file lists.f90.

## 7.106 LISTS::LIST\_ITEM\_IN\_LIST Interface Reference

Determines if an item is in a list and returns the position of the item.

### Private Member Functions

- subroutine [LIST\\_ITEM\\_IN\\_LIST\\_INTG1](#) (LIST, ITEM, LIST\_ITEM, ERR, ERROR,\*)
- subroutine [LIST\\_ITEM\\_IN\\_LIST\\_SP1](#) (LIST, ITEM, LIST\_ITEM, ERR, ERROR,\*)
- subroutine [LIST\\_ITEM\\_IN\\_LIST\\_DP1](#) (LIST, ITEM, LIST\_ITEM, ERR, ERROR,\*)

### 7.106.1 Detailed Description

Determines if an item is in a list and returns the position of the item.

#### See also:

[LISTS](#).

Definition at line 119 of file lists.f90.

### 7.106.2 Member Function Documentation

**7.106.2.1 subroutine LISTS::LIST\_ITEM\_IN\_LIST::LIST\_ITEM\_IN\_LIST\_DP1**  
`(TYPE(LIST_TYPE),pointer LIST, REAL(DP),intent(in) ITEM,  
 INTEGER(INTG),intent(out) LIST_ITEM, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

#### Parameters:

***LIST*** The pointer to the list

***ITEM*** The item to find.

***LIST\_ITEM*** On exit, the position of the item in the list. If the item does not exist then the value of 0 is returned.

***ERR*** The error code

***ERROR*** The error string

Definition at line 707 of file lists.f90.

**7.106.2.2 subroutine LISTS::LIST\_ITEM\_IN\_LIST::LIST\_ITEM\_IN\_LIST\_INTG1**  
`(TYPE(LIST_TYPE),pointer LIST, INTEGER(INTG),intent(in) ITEM,  
 INTEGER(INTG),intent(out) LIST_ITEM, INTEGER(INTG),intent(out) ERR,  
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

#### Parameters:

***LIST*** The pointer to the list

***ITEM*** The item to find.

***LIST\_ITEM*** On exit, the position of the item in the list. If the item does not exist then the value of 0 is returned.

***ERR*** The error code

***ERROR*** The error string.

Definition at line 621 of file lists.f90.

**7.106.2.3 subroutine LISTS::LIST\_ITEM\_IN\_LIST::LIST\_ITEM\_IN\_LIST\_SP1**  
(**TYPE(LIST\_TYPE),pointer LIST,** **REAL(SP),intent(in) ITEM,**  
**INTEGER(INTG),intent(out) LIST\_ITEM,** **INTEGER(INTG),intent(out) ERR,**  
**TYPE(VARYING\_STRING),intent(out) ERROR,** **\*) [private]**

**Parameters:**

***LIST*** The pointer to the list

***ITEM*** The item to find.

***LIST\_ITEM*** On exit, the position of the item in the list. If the item does not exist then the value of 0 is returned.

***ERR*** The error code

***ERROR*** The error string

Definition at line 664 of file lists.f90.

## 7.107 LISTS::LIST\_PTR\_TYPE Struct Reference

Buffer type to allow arrays of pointers to a list.

Collaboration diagram for LISTS::LIST\_PTR\_TYPE:

### Private Attributes

- TYPE([LIST\\_TYPE](#)), pointer PTR

*The pointer to the list.*

### 7.107.1 Detailed Description

Buffer type to allow arrays of pointers to a list.

Definition at line 89 of file lists.f90.

### 7.107.2 Member Data Documentation

#### 7.107.2.1 TYPE(LIST\_TYPE),pointer LISTS::LIST\_PTR\_TYPE::PTR [private]

The pointer to the list.

Definition at line 90 of file lists.f90.

## 7.108 LISTS::LIST\_SEARCH Interface Reference

Searches a list for a given value and returns the position in the list if the value exists.

### Private Member Functions

- subroutine [LIST\\_SEARCH\\_INTG\\_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,\*)
- subroutine [LIST\\_SEARCH\\_SP\\_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,\*)
- subroutine [LIST\\_SEARCH\\_DP\\_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,\*)

### 7.108.1 Detailed Description

Searches a list for a given value and returns the position in the list if the value exists.

**See also:**

[LISTS](#).

Definition at line 133 of file lists.f90.

### 7.108.2 Member Function Documentation

#### 7.108.2.1 subroutine LISTS::LIST\_SEARCH::LIST\_SEARCH\_DP\_ARRAY (REAL(DP),dimension(:),intent(in) A, REAL(DP),intent(in) VALUE, INTEGER(INTG),intent(out) POSITION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]

**Parameters:**

**A** The list to search

**VALUE** The value to search for

**POSITION** On exit, the position of value in the list. If value does not exist in the list the returned position is zero.

**ERR** The error code

**ERROR** The error string

Definition at line 1157 of file lists.f90.

#### 7.108.2.2 subroutine LISTS::LIST\_SEARCH::LIST\_SEARCH\_INTG\_ARRAY (INTEGER(INTG),dimension(:),intent(in) A, INTEGER(INTG),intent(in) VALUE, INTEGER(INTG),intent(out) POSITION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]

**Parameters:**

**A** The list to search

**VALUE** The value to search for

**POSITION** On exit, the position of value in the list. If value does not exist in the list the returned position is zero.

***ERR*** The error code

***ERROR*** The error string

Definition at line 1103 of file lists.f90.

**7.108.2.3 subroutine LISTS::LIST\_SEARCH::LIST\_SEARCH\_SP\_ARRAY**  
**(REAL(SP),dimension(:),intent(in) *A*, REAL(SP),intent(in) *VALUE*,**  
**INTEGER(INTG),intent(out) *POSITION*, INTEGER(INTG),intent(out) *ERR*,**  
**TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

**Parameters:**

***A*** The list to search

***VALUE*** The value to search for

***POSITION*** On exit, the position of value in the list. If value does not exist in the list the returned position is zero.

***ERR*** The error code

***ERROR*** The error string

Definition at line 1130 of file lists.f90.

## 7.109 LISTS::LIST\_SEARCH\_LINEAR Interface Reference

Searches a list using the linear search method.

### Private Member Functions

- subroutine [LIST\\_SEARCH\\_LINEAR\\_INTG\\_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,\*)
- subroutine [LIST\\_SEARCH\\_LINEAR\\_SP\\_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,\*)
- subroutine [LIST\\_SEARCH\\_LINEAR\\_DP\\_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,\*)

### 7.109.1 Detailed Description

Searches a list using the linear search method.

Definition at line 140 of file lists.f90.

### 7.109.2 Member Function Documentation

#### 7.109.2.1 subroutine LISTS::LIST\_SEARCH\_LINEAR::LIST\_SEARCH\_LINEAR\_DP\_ARRAY (REAL(DP),dimension(:),intent(in) A, REAL(DP),intent(in) VALUE, INTEGER(INTG),intent(out) POSITION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]

##### Parameters:

**A** The list to search

**VALUE** The value to search for

**POSITION** On exit, the position of value in the list. If value does not exist in the list the returned position is zero.

**ERR** The error code

**ERROR** The error string

Definition at line 1266 of file lists.f90.

#### 7.109.2.2 subroutine LISTS::LIST\_SEARCH\_LINEAR::LIST\_SEARCH\_LINEAR\_INTG\_- ARRAY (INTEGER(INTG),dimension(:),intent(in) A, INTEGER(INTG),intent(in) VALUE, INTEGER(INTG),intent(out) POSITION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]

##### Parameters:

**A** The list to search

**VALUE** The value to search for

**POSITION** On exit, the position of value in the list. If value does not exist in the list the returned position is zero.

**ERR** The error code

**ERROR** The error string

Definition at line 1184 of file lists.f90.

**7.109.2.3 subroutine LISTS::LIST\_SEARCH\_LINEAR::LIST\_SEARCH\_LINEAR\_SP\_ARRAY**  
(REAL(SP),dimension(:),intent(in) *A*, REAL(SP),intent(in) *VALUE*,  
INTEGER(INTG),intent(out) *POSITION*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

**Parameters:**

*A* The list to search

*VALUE* The value to search for

*POSITION* On exit, the position of value in the list. If value does not exist in the list the returned position is zero.

*ERR* The error code

*ERROR* The error string

Definition at line 1225 of file lists.f90.

## 7.110 LISTS::LIST\_SORT Interface Reference

Sorts a list into ascending order.

### Private Member Functions

- subroutine [LIST\\_SORT\\_INTG\\_ARRAY](#) (A, ERR, ERROR,\*)
- subroutine [LIST\\_SORT\\_SP\\_ARRAY](#) (A, ERR, ERROR,\*)
- subroutine [LIST\\_SORT\\_DP\\_ARRAY](#) (A, ERR, ERROR,\*)

#### 7.110.1 Detailed Description

Sorts a list into ascending order.

Definition at line 147 of file lists.f90.

#### 7.110.2 Member Function Documentation

##### 7.110.2.1 subroutine LISTS::LIST\_SORT::LIST\_SORT\_DP\_ARRAY $(REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR,$ $TYPE(VARYING_STRING),intent(out) ERROR, *)$ [private]

###### Parameters:

- A** The list to sort  
**ERR** The error code  
**ERROR** The error string

Definition at line 1357 of file lists.f90.

##### 7.110.2.2 subroutine LISTS::LIST\_SORT::LIST\_SORT\_INTG\_ARRAY $(INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR,$ $TYPE(VARYING_STRING),intent(out) ERROR, *)$ [private]

###### Parameters:

- A** The list to sort  
**ERR** The error code  
**ERROR** The error string

Definition at line 1307 of file lists.f90.

##### 7.110.2.3 subroutine LISTS::LIST\_SORT::LIST\_SORT\_SP\_ARRAY $(REAL(SP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR,$ $TYPE(VARYING_STRING),intent(out) ERROR, *)$ [private]

###### Parameters:

- A** The list to sort  
**ERR** The error code

***ERROR*** The error string

Definition at line 1332 of file lists.f90.

## 7.111 LISTS::LIST\_SORT\_BUBBLE Interface Reference

Sorts a list into assending order using the bubble sort method.

### Private Member Functions

- subroutine [LIST\\_SORT\\_BUBBLE\\_INTG\\_ARRAY](#) (A, ERR, ERROR,\*)
- subroutine [LIST\\_SORT\\_BUBBLE\\_SP\\_ARRAY](#) (A, ERR, ERROR,\*)
- subroutine [LIST\\_SORT\\_BUBBLE\\_DP\\_ARRAY](#) (A, ERR, ERROR,\*)

#### 7.111.1 Detailed Description

Sorts a list into assending order using the bubble sort method.

Definition at line 154 of file lists.f90.

#### 7.111.2 Member Function Documentation

##### 7.111.2.1 subroutine LISTS::LIST\_SORT\_BUBBLE::LIST\_SORT\_BUBBLE\_DP\_ARRAY $(\text{REAL}(\text{DP}),\text{dimension}(:),\text{intent(inout)} A, \text{ INTEGER}(\text{INTG}),\text{intent(out)} \text{ERR},$ $\text{TYPE}(\text{VARYING\_STRING}),\text{intent(out)} \text{ERROR}, *)$ [private]

###### Parameters:

- A** The list to sort  
**ERR** The error code  
**ERROR** The error string

Definition at line 1463 of file lists.f90.

##### 7.111.2.2 subroutine LISTS::LIST\_SORT\_BUBBLE::LIST\_SORT\_BUBBLE\_INTG\_ARRAY $(\text{INTEGER}(\text{INTG}),\text{dimension}(:),\text{intent(inout)} A, \text{ INTEGER}(\text{INTG}),\text{intent(out)} \text{ERR},$ $\text{TYPE}(\text{VARYING\_STRING}),\text{intent(out)} \text{ERROR}, *)$ [private]

###### Parameters:

- A** The list to sort  
**ERR** The error code  
**ERROR** The error string

Definition at line 1382 of file lists.f90.

##### 7.111.2.3 subroutine LISTS::LIST\_SORT\_BUBBLE::LIST\_SORT\_BUBBLE\_SP\_ARRAY $(\text{REAL}(\text{SP}),\text{dimension}(:),\text{intent(inout)} A, \text{ INTEGER}(\text{INTG}),\text{intent(out)} \text{ERR},$ $\text{TYPE}(\text{VARYING\_STRING}),\text{intent(out)} \text{ERROR}, *)$ [private]

###### Parameters:

- A** The list to sort  
**ERR** The error code

***ERROR*** The error string

Definition at line 1422 of file lists.f90.

## 7.112 LISTS::LIST\_SORT\_HEAP Interface Reference

Sorts a list into assending order using the heap sort method.

### Private Member Functions

- subroutine [LIST\\_SORT\\_HEAP\\_INTG\\_ARRAY](#) (*A, ERR, ERROR,\**)
- subroutine [LIST\\_SORT\\_HEAP\\_SP\\_ARRAY](#) (*A, ERR, ERROR,\**)
- subroutine [LIST\\_SORT\\_HEAP\\_DP\\_ARRAY](#) (*A, ERR, ERROR,\**)

### 7.112.1 Detailed Description

Sorts a list into assending order using the heap sort method.

Definition at line 161 of file lists.f90.

### 7.112.2 Member Function Documentation

#### 7.112.2.1 subroutine LISTS::LIST\_SORT\_HEAP::LIST\_SORT\_HEAP\_DP\_ARRAY (*REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \* [private]*)

##### Parameters:

- A*** The list to sort
- ERR*** The error code
- ERROR*** The error string

Definition at line 1619 of file lists.f90.

#### 7.112.2.2 subroutine LISTS::LIST\_SORT\_HEAP::LIST\_SORT\_HEAP\_INTG\_ARRAY (*INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \* [private]*)

##### Parameters:

- A*** The list to sort
- ERR*** The error code
- ERROR*** The error string

Definition at line 1504 of file lists.f90.

#### 7.112.2.3 subroutine LISTS::LIST\_SORT\_HEAP::LIST\_SORT\_HEAP\_SP\_ARRAY (*REAL(SP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \* [private]*)

##### Parameters:

- A*** The list to sort
- ERR*** The error code

***ERROR*** The error string

Definition at line 1561 of file lists.f90.

## 7.113 LISTS::LIST\_SORT\_SHELL Interface Reference

Sorts a list into either assending or descending order using the shell sort method.

### Private Member Functions

- subroutine [LIST\\_SORT\\_SHELL\\_INTG\\_ARRAY](#) (A, ERR, ERROR,\*)
- subroutine [LIST\\_SORT\\_SHELL\\_SP\\_ARRAY](#) (A, ERR, ERROR,\*)
- subroutine [LIST\\_SORT\\_SHELL\\_DP\\_ARRAY](#) (A, ERR, ERROR,\*)

### 7.113.1 Detailed Description

Sorts a list into either assending or descending order using the shell sort method.

Definition at line 168 of file lists.f90.

### 7.113.2 Member Function Documentation

#### 7.113.2.1 subroutine LISTS::LIST\_SORT\_SHELL::LIST\_SORT\_SHELL\_DP\_ARRAY (REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]

##### Parameters:

**ERR** The error code

**ERROR** The error string

Definition at line 1760 of file lists.f90.

#### 7.113.2.2 subroutine LISTS::LIST\_SORT\_SHELL::LIST\_SORT\_SHELL\_INTG\_ARRAY (INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]

##### Parameters:

**A** The list to sort

**ERR** The error code

**ERROR** The error string

Definition at line 1677 of file lists.f90.

#### 7.113.2.3 subroutine LISTS::LIST\_SORT\_SHELL::LIST\_SORT\_SHELL\_SP\_ARRAY (REAL(SP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]

##### Parameters:

**A** The list to sort

**ERR** The error code

**ERROR** The error string

Definition at line 1718 of file lists.f90.

## 7.114 LISTS::LIST\_TYPE Struct Reference

Contains information on a list.

### Private Attributes

- LOGICAL [LIST\\_FINISHED](#)  
*Is .TRUE. if the list has finished being created, .FALSE. if not.*
- INTEGER(INTG) [NUMBER\\_IN\\_LIST](#)  
*The number of items currently in the list.*
- INTEGER(INTG) [INITIAL\\_SIZE](#)  
*The size of the list when it was initially created.*
- INTEGER(INTG) [SIZE](#)  
*The current size of the list.*
- INTEGER(INTG) [DATA\\_TYPE](#)  
*The data type of the list.*
- INTEGER(INTG) [SORT\\_ORDER](#)  
*The ordering to be used when sorting the list.*
- INTEGER(INTG) [SORT\\_METHOD](#)  
*The sorting method to be used when sorting the list.*
- INTEGER(INTG), pointer [LIST\\_INTG](#)  
*A pointer to the integer data for integer lists.*
- REAL(SP), pointer [LIST\\_SP](#)  
*A pointer to the single precision data for single precision real lists.*
- REAL(DP), pointer [LIST\\_DP](#)  
*A pointer to the double precision data for double precision real lists.*

### 7.114.1 Detailed Description

Contains information on a list.

Definition at line 94 of file lists.f90.

### 7.114.2 Member Data Documentation

#### 7.114.2.1 INTEGER(INTG) LISTS::LIST\_TYPE::DATA\_TYPE [private]

The data type of the list.

See also:

[LISTS::DataType](#)

Definition at line 99 of file lists.f90.

**7.114.2.2 INTEGER(INTG) LISTS::LIST\_TYPE::INITIAL\_SIZE [private]**

The size of the list when it was initially created.

Definition at line 97 of file lists.f90.

**7.114.2.3 REAL(DP),pointer LISTS::LIST\_TYPE::LIST\_DP [private]**

A pointer to the double precision data for double precision real lists.

Definition at line 104 of file lists.f90.

**7.114.2.4 LOGICAL LISTS::LIST\_TYPE::LIST\_FINISHED [private]**

Is .TRUE. if the list has finished being created, .FALSE. if not.

Definition at line 95 of file lists.f90.

**7.114.2.5 INTEGER(INTG),pointer LISTS::LIST\_TYPE::LIST\_INTG [private]**

A pointer to the integer data for integer lists.

Definition at line 102 of file lists.f90.

**7.114.2.6 REAL(SP),pointer LISTS::LIST\_TYPE::LIST\_SP [private]**

A pointer to the single precision data for single precision real lists.

Definition at line 103 of file lists.f90.

**7.114.2.7 INTEGER(INTG) LISTS::LIST\_TYPE::NUMBER\_IN\_LIST [private]**

The number of items currently in the list.

Definition at line 96 of file lists.f90.

**7.114.2.8 INTEGER(INTG) LISTS::LIST\_TYPE::SIZE [private]**

The current size of the list.

Definition at line 98 of file lists.f90.

**7.114.2.9 INTEGER(INTG) LISTS::LIST\_TYPE::SORT\_METHOD [private]**

The sorting method to be used when sorting the list.

**See also:**

[LISTS\\_SortingMethod](#)

Definition at line 101 of file lists.f90.

**7.114.2.10 INTEGER(INTG) LISTS::LIST\_TYPE::SORT\_ORDER [private]**

The ordering to be used when sorting the list.

**See also:**

[LISTS::SortingOrder](#)

Definition at line 100 of file lists.f90.

## 7.115 MATHS::CROSS\_PRODUCT Interface Reference

### Private Member Functions

- subroutine [CROSS\\_PRODUCT\\_INTG](#) (A, B, C, ERR, ERROR,\*)
- subroutine [CROSS\\_PRODUCT\\_SP](#) (A, B, C, ERR, ERROR,\*)
- subroutine [CROSS\\_PRODUCT\\_DP](#) (A, B, C, ERR, ERROR,\*)

#### 7.115.1 Detailed Description

Definition at line 61 of file maths.f90.

#### 7.115.2 Member Function Documentation

**7.115.2.1 subroutine MATHS::CROSS\_PRODUCT::CROSS\_PRODUCT\_DP**  
(REAL(DP),dimension(:),intent(in) *A*, REAL(DP),dimension(:),intent(in) *B*,  
REAL(DP),dimension(:),intent(out) *C*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 250 of file maths.f90.

**7.115.2.2 subroutine MATHS::CROSS\_PRODUCT::CROSS\_PRODUCT\_INTG**  
(INTEGER(INTG),dimension(:),intent(in) *A*, INTEGER(INTG),dimension(:),intent(in)  
*B*, INTEGER(INTG),dimension(:),intent(out) *C*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 162 of file maths.f90.

**7.115.2.3 subroutine MATHS::CROSS\_PRODUCT::CROSS\_PRODUCT\_SP**  
(REAL(SP),dimension(:),intent(in) *A*, REAL(SP),dimension(:),intent(in) *B*,  
REAL(SP),dimension(:),intent(out) *C*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 206 of file maths.f90.

## 7.116 MATHS::D\_CROSS\_PRODUCT Interface Reference

### Private Member Functions

- subroutine [D\\_CROSS\\_PRODUCT\\_INTG](#) (N, A, B, C, D\_A, D\_B, D\_C, ERR, ERROR,\*)
- subroutine [D\\_CROSS\\_PRODUCT\\_SP](#) (N, A, B, C, D\_A, D\_B, D\_C, ERR, ERROR,\*)
- subroutine [D\\_CROSS\\_PRODUCT\\_DP](#) (N, A, B, C, D\_A, D\_B, D\_C, ERR, ERROR,\*)

#### 7.116.1 Detailed Description

Definition at line 67 of file maths.f90.

#### 7.116.2 Member Function Documentation

**7.116.2.1 subroutine MATHS::D\_CROSS\_PRODUCT::D\_CROSS\_PRODUCT\_DP**  
`(INTEGER(INTG),intent(in) N, REAL(DP),dimension(:),intent(in) A,  
REAL(DP),dimension(:),intent(in) B, REAL(DP),dimension(:),intent(out) C,  
REAL(DP),dimension(:, :, ),intent(in) D_A, REAL(DP),dimension(:, :, ),intent(in) D_B,  
REAL(DP),dimension(:, :, ),intent(out) D_C, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Definition at line 415 of file maths.f90.

**7.116.2.2 subroutine MATHS::D\_CROSS\_PRODUCT::D\_CROSS\_PRODUCT\_INTG**  
`(INTEGER(INTG),intent(in) N, INTEGER(INTG),dimension(:),intent(in) A,  
INTEGER(INTG),dimension(:),intent(in) B, INTEGER(INTG),dimension(:),intent(out) C,  
INTEGER(INTG),dimension(:, :, ),intent(in) D_A, INTEGER(INTG),dimension(:, :, ),intent(in) D_B,  
INTEGER(INTG),dimension(:, :, ),intent(out) D_C, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)  
ERROR, *) [private]`

Definition at line 304 of file maths.f90.

**7.116.2.3 subroutine MATHS::D\_CROSS\_PRODUCT::D\_CROSS\_PRODUCT\_SP**  
`(INTEGER(INTG),intent(in) N, REAL(SP),dimension(:),intent(in) A,  
REAL(SP),dimension(:),intent(in) B, REAL(SP),dimension(:),intent(out) C,  
REAL(SP),dimension(:, :, ),intent(in) D_A, REAL(SP),dimension(:, :, ),intent(in) D_B,  
REAL(SP),dimension(:, :, ),intent(out) D_C, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Definition at line 359 of file maths.f90.

## 7.117 MATHS::DETERMINANT Interface Reference

### Private Member Functions

- INTEGER(INTG) [DETERMINANT\\_FULL\\_INTG](#) (A, ERR, ERROR)
- REAL(SP) [DETERMINANT\\_FULL\\_SP](#) (A, ERR, ERROR)
- REAL(DP) [DETERMINANT\\_FULL\\_DP](#) (A, ERR, ERROR)

#### 7.117.1 Detailed Description

Definition at line 73 of file maths.f90.

#### 7.117.2 Member Function Documentation

**7.117.2.1 REAL(DP) MATHS::DETERMINANT::DETERMINANT\_FULL\_DP**  
(REAL(DP),dimension(:,:),intent(in) *A*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]

Definition at line 572 of file maths.f90.

**7.117.2.2 INTEGER(INTG) MATHS::DETERMINANT::DETERMINANT\_FULL\_INTG**  
(INTEGER(INTG),dimension(:,:),intent(in) *A*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]

Definition at line 480 of file maths.f90.

**7.117.2.3 REAL(SP) MATHS::DETERMINANT::DETERMINANT\_FULL\_SP**  
(REAL(SP),dimension(:,:),intent(in) *A*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]

Definition at line 526 of file maths.f90.

## 7.118 MATHS::EDP Interface Reference

### Private Member Functions

- REAL(DP) [EDP\\_DP](#) (X)
- REAL(SP) [EDP\\_SP](#) (X)

#### 7.118.1 Detailed Description

Definition at line 79 of file maths.f90.

#### 7.118.2 Member Function Documentation

##### 7.118.2.1 REAL(DP) MATHS::EDP::EDP\_DP (REAL(DP),intent(in) X) [private]

Definition at line 627 of file maths.f90.

##### 7.118.2.2 REAL(SP) MATHS::EDP::EDP\_SP (REAL(SP),intent(in) X) [private]

Definition at line 663 of file maths.f90.

## 7.119 MATHS::EIGENVALUE Interface Reference

### Private Member Functions

- subroutine [EIGENVALUE\\_FULL\\_SP](#) (A, EVALUES, ERR, ERROR,\*)
- subroutine [EIGENVALUE\\_FULL\\_DP](#) (A, EVALUES, ERR, ERROR,\*)

#### 7.119.1 Detailed Description

Definition at line 84 of file maths.f90.

#### 7.119.2 Member Function Documentation

**7.119.2.1 subroutine MATHS::EIGENVALUE::EIGENVALUE\_FULL\_DP**  
(REAL(DP),dimension(:,:),intent(in) *A*, REAL(DP),dimension(:,),intent(out) *EVALUES*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
\*) [private]

Definition at line 800 of file maths.f90.

**7.119.2.2 subroutine MATHS::EIGENVALUE::EIGENVALUE\_FULL\_SP**  
(REAL(SP),dimension(:,:),intent(in) *A*, REAL(SP),dimension(:,),intent(out) *EVALUES*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
\*) [private]

Definition at line 708 of file maths.f90.

## 7.120 MATHS::EIGENVECTOR Interface Reference

### Private Member Functions

- subroutine [EIGENVECTOR\\_FULL\\_SP](#) (A, EVALUE, EVECTOR, ERR, ERROR,\*)
- subroutine [EIGENVECTOR\\_FULL\\_DP](#) (A, EVALUE, EVECTOR, ERR, ERROR,\*)

#### 7.120.1 Detailed Description

Definition at line 89 of file maths.f90.

#### 7.120.2 Member Function Documentation

**7.120.2.1 subroutine MATHS::EIGENVECTOR::EIGENVECTOR\_FULL\_DP**  
(REAL(DP),dimension(:,:),intent(in) *A*, REAL(DP),intent(in) *EVALUE*,  
REAL(DP),dimension(:,intent(out) *EVECTOR*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 992 of file maths.f90.

**7.120.2.2 subroutine MATHS::EIGENVECTOR::EIGENVECTOR\_FULL\_SP**  
(REAL(SP),dimension(:,:),intent(in) *A*, REAL(SP),intent(in) *EVALUE*,  
REAL(SP),dimension(:,intent(out) *EVECTOR*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 901 of file maths.f90.

## 7.121 MATHS::I0 Interface Reference

### Private Member Functions

- REAL(DP) [I0\\_DP](#) (X)
- REAL(SP) [I0\\_SP](#) (X)

#### 7.121.1 Detailed Description

Definition at line 94 of file maths.f90.

#### 7.121.2 Member Function Documentation

##### 7.121.2.1 **REAL(DP) MATHS::I0::I0\_DP (REAL(DP),intent(in) X) [private]**

Definition at line 1092 of file maths.f90.

##### 7.121.2.2 **REAL(SP) MATHS::I0::I0\_SP (REAL(SP),intent(in) X) [private]**

Definition at line 1127 of file maths.f90.

## 7.122 MATHS::I1 Interface Reference

### Private Member Functions

- REAL(DP) [I1\\_DP](#) (X)
- REAL(SP) [I1\\_SP](#) (X)

#### 7.122.1 Detailed Description

Definition at line 99 of file maths.f90.

#### 7.122.2 Member Function Documentation

##### 7.122.2.1 **REAL(DP) MATHS::I1::I1\_DP (REAL(DP),intent(in) X) [private]**

Definition at line 1171 of file maths.f90.

##### 7.122.2.2 **REAL(SP) MATHS::I1::I1\_SP (REAL(SP),intent(in) X) [private]**

Definition at line 1206 of file maths.f90.

## 7.123 MATHS::INVERT Interface Reference

### Private Member Functions

- subroutine [INVERT\\_FULL\\_SP](#) (A, B, DET, ERR, ERROR,\*)
- subroutine [INVERT\\_FULL\\_DP](#) (A, B, DET, ERR, ERROR,\*)

#### 7.123.1 Detailed Description

Definition at line 104 of file maths.f90.

#### 7.123.2 Member Function Documentation

7.123.2.1 subroutine MATHS::INVERT::INVERT\_FULL\_DP  
(REAL(DP),dimension(:,:),intent(in) *A*, REAL(DP),dimension(:,:),intent(out)  
*B*, REAL(DP),intent(out) *DET*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 1327 of file maths.f90.

7.123.2.2 subroutine MATHS::INVERT::INVERT\_FULL\_SP  
(REAL(SP),dimension(:,:),intent(in) *A*, REAL(SP),dimension(:,:),intent(out)  
*B*, REAL(SP),intent(out) *DET*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 1250 of file maths.f90.

## 7.124 MATHS::K0 Interface Reference

### Private Member Functions

- REAL(DP) [K0\\_DP](#) (X)
- REAL(SP) [K0\\_SP](#) (X)

#### 7.124.1 Detailed Description

Definition at line 109 of file maths.f90.

#### 7.124.2 Member Function Documentation

##### 7.124.2.1 REAL(DP) MATHS::K0::K0\_DP (REAL(DP),intent(in) X) [private]

Definition at line 1413 of file maths.f90.

##### 7.124.2.2 REAL(SP) MATHS::K0::K0\_SP (REAL(SP),intent(in) X) [private]

Definition at line 1464 of file maths.f90.

## 7.125 MATHS::K1 Interface Reference

### Private Member Functions

- REAL(DP) [K1\\_DP](#) (X)
- REAL(SP) [K1\\_SP](#) (X)

#### 7.125.1 Detailed Description

Definition at line 114 of file maths.f90.

#### 7.125.2 Member Function Documentation

##### 7.125.2.1 REAL(DP) MATHS::K1::K1\_DP (REAL(DP),intent(in) X) [private]

Definition at line 1524 of file maths.f90.

##### 7.125.2.2 REAL(SP) MATHS::K1::K1\_SP (REAL(SP),intent(in) X) [private]

Definition at line 1576 of file maths.f90.

## 7.126 MATHS::L2NORM Interface Reference

### Private Member Functions

- REAL(SP) [L2NORM\\_SP](#) (A)
- REAL(DP) [L2NORM\\_DP](#) (A)

#### 7.126.1 Detailed Description

Definition at line 119 of file maths.f90.

#### 7.126.2 Member Function Documentation

##### 7.126.2.1 REAL(DP) MATHS::L2NORM::L2NORM\_DP (REAL(DP),dimension(:),intent(in) A) [private]

Definition at line 1746 of file maths.f90.

##### 7.126.2.2 REAL(SP) MATHS::L2NORM::L2NORM\_SP (REAL(SP),dimension(:),intent(in) A) [private]

Definition at line 1722 of file maths.f90.

## 7.127 MATHS::MATRIX\_PRODUCT Interface Reference

### Private Member Functions

- subroutine [MATRIX\\_PRODUCT\\_SP](#) (A, B, C, ERR, ERROR,\*)
- subroutine [MATRIX\\_PRODUCT\\_DP](#) (A, B, C, ERR, ERROR,\*)

#### 7.127.1 Detailed Description

Definition at line 124 of file maths.f90.

#### 7.127.2 Member Function Documentation

7.127.2.1 subroutine MATHS::MATRIX\_PRODUCT::MATRIX\_PRODUCT\_DP  
(REAL(DP),dimension(3,3),intent(in) *A*, REAL(DP),dimension(3,3),intent(in) *B*,  
REAL(DP),dimension(3,3),intent(out) *C*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 1808 of file maths.f90.

7.127.2.2 subroutine MATHS::MATRIX\_PRODUCT::MATRIX\_PRODUCT\_SP  
(REAL(SP),dimension(3,3),intent(in) *A*, REAL(SP),dimension(3,3),intent(in) *B*,  
REAL(SP),dimension(3,3),intent(out) *C*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 1770 of file maths.f90.

## 7.128 MATHS::MATRIX\_TRANSPOSE Interface Reference

### Private Member Functions

- subroutine [MATRIX\\_TRANSPOSE\\_SP](#) (A, AT, ERR, ERROR,\*)
- subroutine [MATRIX\\_TRANSPOSE\\_DP](#) (A, AT, ERR, ERROR,\*)

#### 7.128.1 Detailed Description

Definition at line 129 of file maths.f90.

#### 7.128.2 Member Function Documentation

7.128.2.1 subroutine MATHS::MATRIX\_TRANSPOSE::MATRIX\_TRANSPOSE\_DP  
(REAL(DP),dimension(3,3),intent(in) A, REAL(DP),dimension(3,3),intent(out) AT,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*) [private]

Definition at line 1881 of file maths.f90.

7.128.2.2 subroutine MATHS::MATRIX\_TRANSPOSE::MATRIX\_TRANSPOSE\_SP  
(REAL(SP),dimension(3,3),intent(in) A, REAL(SP),dimension(3,3),intent(out) AT,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
\*) [private]

Definition at line 1846 of file maths.f90.

## 7.129 MATHS::NORMALISE Interface Reference

### Private Member Functions

- REAL(SP) [NORMALISE\\_SP](#) (A, ERR, ERROR)
- REAL(DP) [NORMALISE\\_DP](#) (A, ERR, ERROR)

#### 7.129.1 Detailed Description

Definition at line 134 of file maths.f90.

#### 7.129.2 Member Function Documentation

##### 7.129.2.1 REAL(DP) MATHS::NORMALISE::NORMALISE\_DP (REAL(DP),dimension(:),intent(in) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]

Definition at line 1962 of file maths.f90.

##### 7.129.2.2 REAL(SP) MATHS::NORMALISE::NORMALISE\_SP (REAL(SP),dimension(:),intent(in) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]

Definition at line 1925 of file maths.f90.

## 7.130 MATHS::SOLVE\_SMALL\_LINEAR\_SYSTEM Interface Reference

### Private Member Functions

- subroutine `SOLVE_SMALL_LINEAR_SYSTEM_SP` (*A*, *x*, *b*, *ERR*, *ERROR*,\*)
- subroutine `SOLVE_SMALL_LINEAR_SYSTEM_DP` (*A*, *x*, *b*, *ERR*, *ERROR*,\*)

#### 7.130.1 Detailed Description

Definition at line 139 of file maths.f90.

#### 7.130.2 Member Function Documentation

**7.130.2.1 subroutine MATHS::SOLVE\_SMALL\_LINEAR\_SYSTEM::SOLVE\_-  
SMALL\_LINEAR\_SYSTEM\_DP** (`REAL(DP),dimension(:,:),intent(in) A,`  
`REAL(DP),dimension(:,intent(out) x,` `REAL(DP),dimension(:,intent(in) b,`  
`INTEGER(INTG),intent(out) ERR,` `TYPE(VARYING_STRING),intent(out) ERROR,`  
\*) [private]

Definition at line 2057 of file maths.f90.

**7.130.2.2 subroutine MATHS::SOLVE\_SMALL\_LINEAR\_SYSTEM::SOLVE\_-  
SMALL\_LINEAR\_SYSTEM\_SP** (`REAL(SP),dimension(:,:),intent(in) A,`  
`REAL(SP),dimension(:,intent(out) x,` `REAL(SP),dimension(:,intent(in) b,`  
`INTEGER(INTG),intent(out) ERR,` `TYPE(VARYING_STRING),intent(out) ERROR,`  
\*) [private]

Definition at line 2008 of file maths.f90.

## 7.131 MATRIX\_VECTOR::MATRIX\_ALL\_VALUES\_SET Interface Reference

### Public Member Functions

- subroutine [MATRIX\\_ALL\\_VALUES\\_SET\\_INTG](#) (MATRIX, VALUE, ERR, ERROR,\*)
- subroutine [MATRIX\\_ALL\\_VALUES\\_SET\\_SP](#) (MATRIX, VALUE, ERR, ERROR,\*)
- subroutine [MATRIX\\_ALL\\_VALUES\\_SET\\_DP](#) (MATRIX, VALUE, ERR, ERROR,\*)
- subroutine [MATRIX\\_ALL\\_VALUES\\_SET\\_L](#) (MATRIX, VALUE, ERR, ERROR,\*)

#### 7.131.1 Detailed Description

Definition at line 178 of file matrix\_vector.f90.

#### 7.131.2 Member Function Documentation

##### 7.131.2.1 subroutine MATRIX\_VECTOR::MATRIX\_ALL\_VALUES\_SET::MATRIX\_ALL\_VALUES\_SET\_DP (TYPE(MATRIX\_TYPE),pointer *MATRIX*, REAL(DP),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

###### Parameters:

*MATRIX* A pointer to the matrix

*VALUE* The value to set

*ERR* The error code

*ERROR* The error string

Definition at line 373 of file matrix\_vector.f90.

##### 7.131.2.2 subroutine MATRIX\_VECTOR::MATRIX\_ALL\_VALUES\_SET::MATRIX\_ALL\_VALUES\_SET\_INTG (TYPE(MATRIX\_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

###### Parameters:

*MATRIX* A pointer to the matrix

*VALUE* The value to set

*ERR* The error code

*ERROR* The error string

Definition at line 293 of file matrix\_vector.f90.

7.131.2.3 subroutine MATRIX\_VECTOR::MATRIX\_ALL\_VALUES\_SET::MATRIX\_ALL\_VALUES\_SET\_L (TYPE(MATRIX\_TYPE),pointer *MATRIX*, LOGICAL,intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:**

*MATRIX* A pointer to the matrix

*VALUE* The value to set

*ERR* The error code

*ERROR* The error string

Definition at line 413 of file matrix\_vector.f90.

7.131.2.4 subroutine MATRIX\_VECTOR::MATRIX\_ALL\_VALUES\_SET::MATRIX\_ALL\_VALUES\_SET\_SP (TYPE(MATRIX\_TYPE),pointer *MATRIX*, REAL(SP),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:**

*MATRIX* A pointer to the matrix

*VALUE* The value to set

*ERR* The error code

*ERROR* The error string

Definition at line 333 of file matrix\_vector.f90.

## 7.132 MATRIX\_VECTOR::MATRIX\_DATA\_GET Interface Reference

### Public Member Functions

- subroutine [MATRIX\\_DATA\\_GET\\_INTG](#) (MATRIX, DATA, ERR, ERROR,\*)
- subroutine [MATRIX\\_DATA\\_GET\\_SP](#) (MATRIX, DATA, ERR, ERROR,\*)
- subroutine [MATRIX\\_DATA\\_GET\\_DP](#) (MATRIX, DATA, ERR, ERROR,\*)
- subroutine [MATRIX\\_DATA\\_GET\\_L](#) (MATRIX, DATA, ERR, ERROR,\*)

#### 7.132.1 Detailed Description

Definition at line 185 of file matrix\_vector.f90.

#### 7.132.2 Member Function Documentation

**7.132.2.1 subroutine MATRIX\_VECTOR::MATRIX\_DATA\_GET::MATRIX\_DATA\_GET\_DP**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, REAL(DP),dimension(:),pointer DATA,  
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,  
 *)`

##### Parameters:

*MATRIX* A pointer to the matrix

*DATA* On return a pointer to the matrix data

*ERR* The error code

*ERROR* The error string

Definition at line 712 of file matrix\_vector.f90.

**7.132.2.2 subroutine MATRIX\_VECTOR::MATRIX\_DATA\_GET::MATRIX\_DATA\_GET\_INTG**  
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),pointer  
DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)  
ERROR, *)`

##### Parameters:

*MATRIX* A pointer to the matrix

*DATA* On return a pointer to the matrix data

*ERR* The error code

*ERROR* The error string

Definition at line 622 of file matrix\_vector.f90.

7.132.2.3 subroutine MATRIX\_VECTOR::MATRIX\_DATA\_GET::MATRIX\_DATA\_GET\_L  
(TYPE(MATRIX\_TYPE),pointer *MATRIX*, LOGICAL,dimension(:),pointer *DATA*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
\*)

**Parameters:**

*MATRIX* A pointer to the matrix  
*DATA* On return a pointer to the matrix data  
*ERR* The error code  
*ERROR* The error string

Definition at line 757 of file matrix\_vector.f90.

7.132.2.4 subroutine MATRIX\_VECTOR::MATRIX\_DATA\_GET::MATRIX\_DATA\_GET\_SP  
(TYPE(MATRIX\_TYPE),pointer *MATRIX*, REAL(SP),dimension(:),pointer *DATA*,  
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*,  
\*)

**Parameters:**

*MATRIX* A pointer to the matrix  
*DATA* On return a pointer to the matrix data  
*ERR* The error code  
*ERROR* The error string

Definition at line 667 of file matrix\_vector.f90.

## 7.133 MATRIX\_VECTOR::MATRIX\_VALUES\_ADD Interface Reference

### Public Member Functions

- subroutine `MATRIX_VALUES_ADD_INTG` (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine `MATRIX_VALUES_ADD_INTG1` (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
- subroutine `MATRIX_VALUES_ADD_INTG2` (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine `MATRIX_VALUES_ADD_SP` (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine `MATRIX_VALUES_ADD_SP1` (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
- subroutine `MATRIX_VALUES_ADD_SP2` (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine `MATRIX_VALUES_ADD_DP` (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine `MATRIX_VALUES_ADD_DP1` (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
- subroutine `MATRIX_VALUES_ADD_DP2` (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine `MATRIX_VALUES_ADD_L` (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine `MATRIX_VALUES_ADD_L1` (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
- subroutine `MATRIX_VALUES_ADD_L2` (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)

### 7.133.1 Detailed Description

Definition at line 192 of file matrix\_vector.f90.

### 7.133.2 Member Function Documentation

**7.133.2.1 subroutine `MATRIX_VECTOR::MATRIX_VALUES_ADD::MATRIX_VALUES_ADD_DP`** (TYPE(MATRIX\_TYPE),pointer *MATRIX*,  
 INTEGER(INTG),dimension(:),intent(in) *ROW\_INDICES*,  
 INTEGER(INTG),dimension(:),intent(in) *COLUMN\_INDICES*,  
 REAL(DP),dimension(:),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

#### Parameters:

***MATRIX*** A pointer to the matrix

***ROW\_INDICES*** *ROW\_INDICES*(i). The row index for the i'th value to add

***COLUMN\_INDICES*** *COLUMN\_INDICES*(i). The column index for the i'th value to add

***VALUES*** *VALUES*(i). The value of the i'th value to add

***ERR*** The error code

**ERROR** The error string

Definition at line 2166 of file matrix\_vector.f90.

**7.133.2.2 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD::MATRIX\_VALUES\_ADD\_DP1** (TYPE(MATRIX\_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(in) *ROW\_INDEX*, INTEGER(INTG),intent(in) *COLUMN\_INDEX*, REAL(DP),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:**

*MATRIX* A pointer to the matrix

*ROW\_INDEX* The row index for the value to add

*COLUMN\_INDEX* The column index for the value to add

*VALUE* The value to add

*ERR* The error code

*ERROR* The error string

Definition at line 2232 of file matrix\_vector.f90.

**7.133.2.3 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD::MATRIX\_VALUES\_ADD\_DP2** (TYPE(MATRIX\_TYPE),pointer *MATRIX*, INTEGER(INTG),dimension(:,),intent(in) *ROW\_INDICES*, INTEGER(INTG),dimension(:,),intent(in) *COLUMN\_INDICES*, REAL(DP),dimension(:, :,),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:**

*MATRIX* A pointer to the matrix

*ROW\_INDICES* ROW\_INDICES(i). The row index for the ij'th value to add

*COLUMN\_INDICES* COLUMN\_INDICES(j). The column index for the ij'th value to add

*VALUES* VALUES(i,j). The value of the ij'th value to add

*ERR* The error code

*ERROR* The error string

Definition at line 2282 of file matrix\_vector.f90.

**7.133.2.4 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD::MATRIX\_VALUES\_ADD\_INTG** (TYPE(MATRIX\_TYPE),pointer *MATRIX*, INTEGER(INTG),dimension(:,),intent(in) *ROW\_INDICES*, INTEGER(INTG),dimension(:,),intent(in) *COLUMN\_INDICES*, INTEGER(INTG),dimension(:,),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:**

*MATRIX* A pointer to the matrix

**ROW\_INDICES** ROW\_INDICES(i). The row index for the i'th value to add  
**COLUMN\_INDICES** COLUMN\_INIDICES(i). The column index for the i'th value to add  
**VALUES** VALUES(i). The value of the i'th value to add  
**ERR** The error code  
**ERROR** The error string

Definition at line 1794 of file matrix\_vector.f90.

**7.133.2.5 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD::MATRIX\_VALUES\_ADD\_INTG1** (TYPE(MATRIX\_TYPE),pointer **MATRIX**,  
 INTEGER(INTG),intent(in) **ROW\_INDEX**, INTEGER(INTG),intent(in)  
**COLUMN\_INDEX**, INTEGER(INTG),intent(in) **VALUE**, INTEGER(INTG),intent(out)  
**ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

**Parameters:**

**MATRIX** A pointer to the matrix  
**ROW\_INDEX** The row index for the value to add  
**COLUMN\_INDEX** The column index for the value to add  
**VALUE** The value to add  
**ERR** The error code  
**ERROR** The error string

Definition at line 1860 of file matrix\_vector.f90.

**7.133.2.6 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD::MATRIX\_VALUES\_ADD\_INTG2** (TYPE(MATRIX\_TYPE),pointer **MATRIX**,  
 INTEGER(INTG),dimension(:,),intent(in) **ROW\_INDICES**,  
 INTEGER(INTG),dimension(:,),intent(in) **COLUMN\_INDICES**,  
 INTEGER(INTG),dimension(:,:,),intent(in) **VALUES**, INTEGER(INTG),intent(out)  
**ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

**Parameters:**

**MATRIX** A pointer to the matrix  
**ROW\_INDICES** ROW\_INDICES(i). The row index for the ij'th value to add  
**COLUMN\_INDICES** COLUMN\_INIDICES(j). The column index for the ij'th value to add  
**VALUES** VALUES(i,j). The value of the ij'th value to add  
**ERR** The error code  
**ERROR** The error string

Definition at line 1910 of file matrix\_vector.f90.

---

**7.133.2.7 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD::MATRIX\_VALUES\_ADD\_L (TYPE(MATRIX\_TYPE),pointer *MATRIX*, INTEGER(INTG),dimension(:),intent(in) *ROW\_INDICES*, INTEGER(INTG),dimension(:),intent(in) *COLUMN\_INDICES*, LOGICAL,dimension(:),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

**Parameters:*****MATRIX*** A pointer to the matrix***ROW\_INDICES*** *ROW\_INDICES*(i). The row index for the i'th value to add***COLUMN\_INDICES*** *COLUMN\_INIDICES*(i). The column index for the i'th value to add***VALUES*** *VALUES*(i). The value of the i'th value to add***ERR*** The error code***ERROR*** The error string

Definition at line 2352 of file matrix\_vector.f90.

**7.133.2.8 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD::MATRIX\_VALUES\_ADD\_L1 (TYPE(MATRIX\_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(in) *ROW\_INDEX*, INTEGER(INTG),intent(in) *COLUMN\_INDEX*, LOGICAL,intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

**Parameters:*****MATRIX*** A pointer to the matrix***ROW\_INDEX*** The row index for the value to add***COLUMN\_INDEX*** The column index for the value to add***VALUE*** The value to add***ERR*** The error code***ERROR*** The error string

Definition at line 2418 of file matrix\_vector.f90.

**7.133.2.9 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD::MATRIX\_VALUES\_ADD\_L2 (TYPE(MATRIX\_TYPE),pointer *MATRIX*, INTEGER(INTG),dimension(:),intent(in) *ROW\_INDICES*, INTEGER(INTG),dimension(:),intent(in) *COLUMN\_INDICES*, LOGICAL,dimension(:, :, ),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

**Parameters:*****MATRIX*** A pointer to the matrix***ROW\_INDICES*** *ROW\_INDICES*(i). The row index for the ij'th value to add***COLUMN\_INDICES*** *COLUMN\_INIDICES*(j). The column index for the ij'th value to add***VALUES*** *VALUES*(i,j). The value of the ij'th value to add***ERR*** The error code***ERROR*** The error string

Definition at line 2468 of file matrix\_vector.f90.

---

**7.133.2.10 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD::MATRIX\_VALUES\_ADD\_SP** (TYPE(MATRIX\_TYPE),pointer *MATRIX*,  
 INTEGER(INTG),dimension(:),intent(in) *ROW\_INDICES*,  
 INTEGER(INTG),dimension(:),intent(in) *COLUMN\_INDICES*,  
 REAL(SP),dimension(:),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:*****MATRIX*** A pointer to the matrix***ROW\_INDICES*** *ROW\_INDICES*(i). The row index for the i'th value to add***COLUMN\_INDICES*** *COLUMN\_INIDICES*(i). The column index for the i'th value to add***VALUES*** *VALUES*(i). The value of the i'th value to add***ERR*** The error code***ERROR*** The error string

Definition at line 1980 of file matrix\_vector.f90.

**7.133.2.11 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD::MATRIX\_VALUES\_ADD\_SP1** (TYPE(MATRIX\_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(in)  
*ROW\_INDEX*, INTEGER(INTG),intent(in) *COLUMN\_INDEX*, REAL(SP),intent(in) *VALUE*,  
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:*****MATRIX*** A pointer to the matrix***ROW\_INDEX*** The row index for the value to add***COLUMN\_INDEX*** The column index for the value to add***VALUE*** The value to add***ERR*** The error code***ERROR*** The error string

Definition at line 2046 of file matrix\_vector.f90.

**7.133.2.12 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_ADD::MATRIX\_VALUES\_ADD\_SP2** (TYPE(MATRIX\_TYPE),pointer *MATRIX*,  
 INTEGER(INTG),dimension(:),intent(in) *ROW\_INDICES*,  
 INTEGER(INTG),dimension(:),intent(in) *COLUMN\_INDICES*,  
 REAL(SP),dimension(:, :),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:*****MATRIX*** A pointer to the matrix***ROW\_INDICES*** *ROW\_INDICES*(i). The row index for the ij'th value to add***COLUMN\_INDICES*** *COLUMN\_INIDICES*(j). The column index for the ij'th value to add***VALUES*** *VALUES*(i,j). The value of the ij'th value to add***ERR*** The error code***ERROR*** The error string

Definition at line 2096 of file matrix\_vector.f90.

## 7.134 MATRIX\_VECTOR::MATRIX\_VALUES\_GET Interface Reference

### Public Member Functions

- subroutine [MATRIX\\_VALUES\\_GET\\_INTG](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_GET\\_INTG1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_GET\\_INTG2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_GET\\_SP](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_GET\\_SP1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_GET\\_SP2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_GET\\_DP](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_GET\\_DP1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_GET\\_DP2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_GET\\_L](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_GET\\_L1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_GET\\_L2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)

#### 7.134.1 Detailed Description

Definition at line 207 of file matrix\_vector.f90.

#### 7.134.2 Member Function Documentation

**7.134.2.1 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_GET::MATRIX\_VALUES\_GET\_DP** (TYPE(MATRIX\_TYPE),pointer *MATRIX*,  
 INTEGER(INTG),dimension(:),intent(in) *ROW\_INDICES*,  
 INTEGER(INTG),dimension(:),intent(in) *COLUMN\_INDICES*,  
 REAL(DP),dimension(:),intent(out) *VALUES*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

##### Parameters:

***MATRIX*** A pointer to the matrix

***ROW\_INDICES*** *ROW\_INDICES*(i). The row index for the i'th value to get

***COLUMN\_INDICES*** *COLUMN\_INDICES*(i). The column index for the i'th value to get

***VALUES*** *VALUES*(i). On return the value of the i'th value to get

***ERR*** The error code

**ERROR** The error string

Definition at line 2898 of file matrix\_vector.f90.

**7.134.2.2 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_GET::MATRIX\_VALUES\_-  
GET\_DP1** (TYPE(MATRIX\_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(in)  
*ROW\_INDEX*, INTEGER(INTG),intent(in) *COLUMN\_INDEX*, REAL(DP),intent(out)  
*VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out)  
*ERROR*, \*)

**Parameters:**

*MATRIX* A pointer to the matrix

*ROW\_INDEX* The row index of the value to get

*COLUMN\_INDEX* The column index of the value to get

*VALUE* On return the value in the matrix at the specified row and column

*ERR* The error code

*ERROR* The error string

Definition at line 2962 of file matrix\_vector.f90.

**7.134.2.3 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_GET::MATRIX\_-  
VALUES\_GET\_DP2** (TYPE(MATRIX\_TYPE),pointer *MATRIX*,  
INTEGER(INTG),dimension(:,),intent(in) *ROW\_INDICES*,  
INTEGER(INTG),dimension(:,),intent(in) *COLUMN\_INDICES*,  
REAL(DP),dimension(:, :, ),intent(out) *VALUES*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:**

*MATRIX* A pointer to the matrix

*ROW\_INDICES* *ROW\_INDICES*(i). The row index for the ij'th value to get

*COLUMN\_INDICES* *COLUMN\_INDICES*(j). The column index for the ij'th value to get

*VALUES* *VALUES*(i,j). On return the value of the ij'th value to get

*ERR* The error code

*ERROR* The error string

Definition at line 3010 of file matrix\_vector.f90.

**7.134.2.4 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_GET::MATRIX\_-  
VALUES\_GET\_INTG** (TYPE(MATRIX\_TYPE),pointer *MATRIX*,  
INTEGER(INTG),dimension(:,),intent(in) *ROW\_INDICES*,  
INTEGER(INTG),dimension(:,),intent(in) *COLUMN\_INDICES*,  
INTEGER(INTG),dimension(:,),intent(out) *VALUES*, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:**

*MATRIX* A pointer to the matrix

**ROW\_INDICES** ROW\_INDICES(i). The row index for the i'th value to get

**COLUMN\_INDICES** COLUMN\_INIDICES(i). The column index for the i'th value to get

**VALUES** VALUES(i). On return the value of the i'th value to get

**ERR** The error code

**ERROR** The error string

Definition at line 2538 of file matrix\_vector.f90.

**7.134.2.5 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_GET::MATRIX\_VALUES\_-  
GET\_INTG1 (TYPE(MATRIX\_TYPE),pointer MATRIX, INTEGER(INTG),intent(in)  
ROW\_INDEX, INTEGER(INTG),intent(in) COLUMN\_INDEX,  
INTEGER(INTG),intent(out) VALUE, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

**Parameters:**

**MATRIX** A pointer to the matrix

**ROW\_INDEX** The row index of the value to get

**COLUMN\_INDEX** The column index of the value to get

**VALUE** On return the value in the matrix at the specified row and column

**ERR** The error code

**ERROR** The error string

Definition at line 2602 of file matrix\_vector.f90.

**7.134.2.6 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_GET::MATRIX\_-  
VALUES\_GET\_INTG2 (TYPE(MATRIX\_TYPE),pointer MATRIX,  
INTEGER(INTG),dimension(:),intent(in) ROW\_INDICES,  
INTEGER(INTG),dimension(:),intent(in) COLUMN\_INDICES,  
INTEGER(INTG),dimension(:,:,),intent(out) VALUES, INTEGER(INTG),intent(out)  
ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

**Parameters:**

**MATRIX** A pointer to the matrix

**ROW\_INDICES** ROW\_INDICES(i). The row index for the ij'th value to get

**COLUMN\_INDICES** COLUMN\_INIDICES(j). The column index for the ij'th value to get

**VALUES** VALUES(i,j). On return the value of the ij'th value to get

**ERR** The error code

**ERROR** The error string

Definition at line 2650 of file matrix\_vector.f90.

---

**7.134.2.7 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_GET::MATRIX\_VALUES\_GET\_L (TYPE(MATRIX\_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW\_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN\_INDICES, LOGICAL,dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

**Parameters:****MATRIX** A pointer to the matrix**ROW\_INDICES** ROW\_INDICES(i). The row index for the i'th value to get**COLUMN\_INDICES** COLUMN\_INIDICES(i). The column index for the i'th value to get**VALUES** VALUES(i). On return the value of the i'th value to get**ERR** The error code**ERROR** The error string

Definition at line 3078 of file matrix\_vector.f90.

**7.134.2.8 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_GET::MATRIX\_VALUES\_GET\_L1 (TYPE(MATRIX\_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW\_INDEX, INTEGER(INTG),intent(in) COLUMN\_INDEX, LOGICAL,intent(out) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

**Parameters:****MATRIX** A pointer to the matrix**ROW\_INDEX** The row index of the value to get**COLUMN\_INDEX** The column index of the value to get**VALUE** On return the value in the matrix at the specified row and column**ERR** The error code**ERROR** The error string

Definition at line 3142 of file matrix\_vector.f90.

**7.134.2.9 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_GET::MATRIX\_VALUES\_GET\_L2 (TYPE(MATRIX\_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW\_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN\_INDICES, LOGICAL,dimension(:, :, intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

**Parameters:****MATRIX** A pointer to the matrix**ROW\_INDICES** ROW\_INDICES(i). The row index for the ij'th value to get**COLUMN\_INDICES** COLUMN\_INIDICES(j). The column index for the ij'th value to get**VALUES** VALUES(i,j). On return the value of the ij'th value to get**ERR** The error code**ERROR** The error string

Definition at line 3190 of file matrix\_vector.f90.

---

**7.134.2.10 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_GET::MATRIX\_VALUES\_GET\_SP (TYPE(MATRIX\_TYPE),pointer *MATRIX*, INTEGER(INTG),dimension(:),intent(in) *ROW\_INDICES*, INTEGER(INTG),dimension(:),intent(in) *COLUMN\_INDICES*, REAL(SP),dimension(:),intent(out) *VALUES*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

**Parameters:*****MATRIX*** A pointer to the matrix***ROW\_INDICES*** *ROW\_INDICES*(i). The row index for the i'th value to get***COLUMN\_INDICES*** *COLUMN\_INIDICES*(i). The column index for the i'th value to get***VALUES*** *VALUES*(i). On return the value of the i'th value to get***ERR*** The error code***ERROR*** The error string

Definition at line 2718 of file matrix\_vector.f90.

**7.134.2.11 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_GET::MATRIX\_VALUES\_GET\_SP1 (TYPE(MATRIX\_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(in) *ROW\_INDEX*, INTEGER(INTG),intent(in) *COLUMN\_INDEX*, REAL(SP),intent(out) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

**Parameters:*****MATRIX*** A pointer to the matrix***ROW\_INDEX*** The row index of the value to get***COLUMN\_INDEX*** The column index of the value to get***VALUE*** On return the value in the matrix at the specified row and column***ERR*** The error code***ERROR*** The error string

Definition at line 2782 of file matrix\_vector.f90.

**7.134.2.12 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_GET::MATRIX\_VALUES\_GET\_SP2 (TYPE(MATRIX\_TYPE),pointer *MATRIX*, INTEGER(INTG),dimension(:),intent(in) *ROW\_INDICES*, INTEGER(INTG),dimension(:),intent(in) *COLUMN\_INDICES*, REAL(SP),dimension(:, :),intent(out) *VALUES*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)**

**Parameters:*****MATRIX*** A pointer to the matrix***ROW\_INDICES*** *ROW\_INDICES*(i). The row index for the ij'th value to get***COLUMN\_INDICES*** *COLUMN\_INIDICES*(j). The column index for the ij'th value to get***VALUES*** *VALUES*(i,j). On return the value of the ij'th value to get***ERR*** The error code***ERROR*** The error string

Definition at line 2830 of file matrix\_vector.f90.

## 7.135 MATRIX\_VECTOR::MATRIX\_VALUES\_SET Interface Reference

### Public Member Functions

- subroutine [MATRIX\\_VALUES\\_SET\\_INTG](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_SET\\_INTG1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_SET\\_INTG2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_SET\\_SP](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_SET\\_SP1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_SET\\_SP2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_SET\\_DP](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_SET\\_DP1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_SET\\_DP2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_SET\\_L](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_SET\\_L1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
- subroutine [MATRIX\\_VALUES\\_SET\\_L2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)

#### 7.135.1 Detailed Description

Definition at line 222 of file matrix\_vector.f90.

#### 7.135.2 Member Function Documentation

**7.135.2.1 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_SET::MATRIX\_VALUES\_SET\_DP** (TYPE(MATRIX\_TYPE),pointer **MATRIX**,  
**INTEGER(INTG),dimension(:),intent(in) *ROW\_INDICES***,  
**INTEGER(INTG),dimension(:),intent(in) *COLUMN\_INDICES***,  
**REAL(DP),dimension(:),intent(in) *VALUES***, **INTEGER(INTG),intent(out) *ERR***,  
**TYPE(VARYING\_STRING),intent(out) *ERROR***, \*)

##### Parameters:

**MATRIX** A pointer to the matrix to set.

**ROW\_INDICES** *ROW\_INDICES*(i). The row index of the i'th value to set

**COLUMN\_INDICES** *COLUMN\_INDICES*(i). The column index of the i'th value to set

**VALUES** *VALUES*(i). The value of the i'th value to set

**ERR** The error code

**ERROR** The error string

Definition at line 3630 of file matrix\_vector.f90.

**7.135.2.2 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_SET::MATRIX\_VALUES\_SET\_DP1** (TYPE(MATRIX\_TYPE),pointer **MATRIX**, INTEGER(INTG),intent(in) **ROW\_INDEX**, INTEGER(INTG),intent(in) **COLUMN\_INDEX**, REAL(DP),intent(in) **VALUE**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

**Parameters:**

**MATRIX** A pointer to the matrix

**ROW\_INDEX** The row index of the value to set

**COLUMN\_INDEX** The column index of the value to set

**VALUE** The value to set

**ERR** The error code

**ERROR** The error string

Definition at line 3696 of file matrix\_vector.f90.

**7.135.2.3 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_SET::MATRIX\_VALUES\_SET\_DP2** (TYPE(MATRIX\_TYPE),pointer **MATRIX**, INTEGER(INTG),dimension(:,),intent(in) **ROW\_INDICES**, INTEGER(INTG),dimension(:,),intent(in) **COLUMN\_INDICES**, REAL(DP),dimension(:, :,),intent(in) **VALUES**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

**Parameters:**

**MATRIX** A pointer to the matrix to set.

**ROW\_INDICES** ROW\_INDICES(i). The row index of the ij'th value to set

**COLUMN\_INDICES** COLUMN\_INDICES(j). The column index of the ij'th value to set

**VALUES** VALUES(i,j). The value of the ij'th value to set

**ERR** The error code

**ERROR** The error string

Definition at line 3746 of file matrix\_vector.f90.

**7.135.2.4 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_SET::MATRIX\_VALUES\_SET\_INTG** (TYPE(MATRIX\_TYPE),pointer **MATRIX**, INTEGER(INTG),dimension(:,),intent(in) **ROW\_INDICES**, INTEGER(INTG),dimension(:,),intent(in) **COLUMN\_INDICES**, INTEGER(INTG),dimension(:,),intent(in) **VALUES**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

**Parameters:**

**MATRIX** A pointer to the matrix

**ROW\_INDICES** ROW\_INDICES(i). The row index for the i'th value to set  
**COLUMN\_INDICES** COLUMN\_INIDICES(i). The column index for the i'th value to set  
**VALUES** VALUES(i). The value of the i'th value to set  
**ERR** The error code  
**ERROR** The error string

Definition at line 3258 of file matrix\_vector.f90.

**7.135.2.5 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_SET::MATRIX\_VALUES\_SET\_INTG1** (TYPE(MATRIX\_TYPE),pointer *MATRIX*,  
 INTEGER(INTG),intent(in) *ROW\_INDEX*, INTEGER(INTG),intent(in)  
*COLUMN\_INDEX*, INTEGER(INTG),intent(in) *VALUE*, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:**

**MATRIX** A pointer to the matrix  
**ROW\_INDEX** The row index of the value to set  
**COLUMN\_INDEX** The column index of the value to set  
**VALUE** The value to set  
**ERR** The error code  
**ERROR** The error string

Definition at line 3324 of file matrix\_vector.f90.

**7.135.2.6 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_SET::MATRIX\_VALUES\_SET\_INTG2** (TYPE(MATRIX\_TYPE),pointer *MATRIX*,  
 INTEGER(INTG),dimension(:,),intent(in) *ROW\_INDICES*,  
 INTEGER(INTG),dimension(:,),intent(in) *COLUMN\_INDICES*,  
 INTEGER(INTG),dimension(:, :, ),intent(in) *VALUES*, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:**

**MATRIX** A pointer to the matrix  
**ROW\_INDICES** ROW\_INDICES(i). The row index for the ij'th value to set  
**COLUMN\_INDICES** COLUMN\_INIDICES(j). The column index for the ij'th value to set  
**VALUES** VALUES(i,j). The value of the i,j'th value to set  
**ERR** The error code  
**ERROR** The error string

Definition at line 3374 of file matrix\_vector.f90.

**7.135.2.7 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_SET::MATRIX\_VALUES\_SET\_L** (TYPE(MATRIX\_TYPE),pointer *MATRIX*, INTEGER(INTG),dimension(:,),intent(in)  
*ROW\_INDICES*, INTEGER(INTG),dimension(:,),intent(in) *COLUMN\_INDICES*,  
 LOGICAL,dimension(:,),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:**

**MATRIX** A pointer to the matrix to set

**ROW\_INDICES** ROW\_INDICES(i). The row index of the i'th value to set

**COLUMN\_INDICES** COLUMN\_INDICES(i). The column index of the i'th value to set

**VALUES** VALUES(i). The value of the i'th value to set

**ERR** The error code

**ERROR** The error string

Definition at line 3816 of file matrix\_vector.f90.

**7.135.2.8 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_SET::MATRIX\_VALUES\_SET\_L1** (TYPE(MATRIX\_TYPE),pointer **MATRIX**, INTEGER(INTG),intent(in) **ROW\_INDEX**, INTEGER(INTG),intent(in) **COLUMN\_INDEX**, LOGICAL,intent(in) **VALUE**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

**Parameters:**

**MATRIX** A pointer to the matrix

**ROW\_INDEX** The row index of the value to set

**COLUMN\_INDEX** The column index of the value to set

**VALUE** The value to set

**ERR** The error code

**ERROR** The error string

Definition at line 3882 of file matrix\_vector.f90.

**7.135.2.9 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_SET::MATRIX\_VALUES\_SET\_L2** (TYPE(MATRIX\_TYPE),pointer **MATRIX**, INTEGER(INTG),dimension(:,),intent(in) **ROW\_INDICES**, INTEGER(INTG),dimension(:,),intent(in) **COLUMN\_INDICES**, LOGICAL,dimension(:, :,),intent(in) **VALUES**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

**Parameters:**

**MATRIX** A pointer to the matrix to set

**ROW\_INDICES** ROW\_INDICES(i). The row index of the ij'th value to set

**COLUMN\_INDICES** COLUMN\_INDICES(j). The column index of the ij'th value to set

**VALUES** VALUES(i,j). The value of the ij'th value to set

**ERR** The error code

**ERROR** The error string

Definition at line 3932 of file matrix\_vector.f90.

---

**7.135.2.10 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_SET::MATRIX\_VALUES\_SET\_SP** (TYPE(MATRIX\_TYPE),pointer *MATRIX*,  
 INTEGER(INTG),dimension(:),intent(in) *ROW\_INDICES*,  
 INTEGER(INTG),dimension(:),intent(in) *COLUMN\_INDICES*,  
 REAL(SP),dimension(:),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:*****MATRIX*** A pointer to the matrix to set***ROW\_INDICES*** *ROW\_INDICES*(i). The row index of the i'th value to set***COLUMN\_INDICES*** *COLUMN\_INDICES*(i). The column index of the i'th value to set***VALUES*** *VALUES*(i). The value of the i'th value to set***ERR*** The error code***ERROR*** The error string

Definition at line 3444 of file matrix\_vector.f90.

**7.135.2.11 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_SET::MATRIX\_VALUES\_SET\_SP1** (TYPE(MATRIX\_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(in)  
*ROW\_INDEX*, INTEGER(INTG),intent(in) *COLUMN\_INDEX*, REAL(SP),intent(in)  
*VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out)  
*ERROR*, \*)

**Parameters:*****MATRIX*** A pointer to the matrix***ROW\_INDEX*** The row index of the value to set***COLUMN\_INDEX*** The column index of the value to set***VALUE*** The value to set***ERR*** The error code***ERROR*** The error string

Definition at line 3510 of file matrix\_vector.f90.

**7.135.2.12 subroutine MATRIX\_VECTOR::MATRIX\_VALUES\_SET::MATRIX\_VALUES\_SET\_SP2** (TYPE(MATRIX\_TYPE),pointer *MATRIX*,  
 INTEGER(INTG),dimension(:),intent(in) *ROW\_INDICES*,  
 INTEGER(INTG),dimension(:),intent(in) *COLUMN\_INDICES*,  
 REAL(SP),dimension(:, :),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*,  
 TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:*****MATRIX*** A pointer to the matrix to set***ROW\_INDICES*** *ROW\_INDICES*(i). The row index of the ij'th value to set***COLUMN\_INDICES*** *COLUMN\_INDICES*(j). The column index of the ij'th value to set***VALUES*** *VALUES*(i,j). The value of the ij'th value to set***ERR*** The error code***ERROR*** The error string

Definition at line 3560 of file matrix\_vector.f90.

## 7.136 MATRIX\_VECTOR::VECTOR\_ALL\_VALUES\_SET Interface Reference

### Public Member Functions

- subroutine [VECTOR\\_ALL\\_VALUES\\_SET\\_INTG](#) (VECTOR, VALUE, ERR, ERROR,\*)
- subroutine [VECTOR\\_ALL\\_VALUES\\_SET\\_SP](#) (VECTOR, VALUE, ERR, ERROR,\*)
- subroutine [VECTOR\\_ALL\\_VALUES\\_SET\\_DP](#) (VECTOR, VALUE, ERR, ERROR,\*)
- subroutine [VECTOR\\_ALL\\_VALUES\\_SET\\_L](#) (VECTOR, VALUE, ERR, ERROR,\*)

#### 7.136.1 Detailed Description

Definition at line 237 of file matrix\_vector.f90.

#### 7.136.2 Member Function Documentation

**7.136.2.1 subroutine MATRIX\_VECTOR::VECTOR\_ALL\_VALUES\_SET::VECTOR\_ALL\_VALUES\_SET\_DP** (TYPE(VECTOR\_TYPE),pointer **VECTOR**, REAL(DP),intent(in) **VALUE**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

##### Parameters:

**VECTOR** A pointer to the vector to set

**VALUE** The value to set the vector to

**ERR** The error code

**ERROR** The error string

Definition at line 4082 of file matrix\_vector.f90.

**7.136.2.2 subroutine MATRIX\_VECTOR::VECTOR\_ALL\_VALUES\_SET::VECTOR\_ALL\_VALUES\_SET\_INTG** (TYPE(VECTOR\_TYPE),pointer **VECTOR**, INTEGER(INTG),intent(in) **VALUE**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

##### Parameters:

**VECTOR** A pointer to the vector to set

**VALUE** The value to set the vector to

**ERR** The error code

**ERROR** The error string

Definition at line 4002 of file matrix\_vector.f90.

7.136.2.3 subroutine MATRIX\_VECTOR::VECTOR\_ALL\_VALUES\_SET::VECTOR\_ALL\_VALUES\_SET\_L (TYPE(VECTOR\_TYPE),pointer **VECTOR**, LOGICAL,intent(in) **VALUE**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

**Parameters:**

**VECTOR** A pointer to the vector to set  
**VALUE** The value to set the vector to  
**ERR** The error code  
**ERROR** The error string

Definition at line 4122 of file matrix\_vector.f90.

7.136.2.4 subroutine MATRIX\_VECTOR::VECTOR\_ALL\_VALUES\_SET::VECTOR\_ALL\_VALUES\_SET\_SP (TYPE(VECTOR\_TYPE),pointer **VECTOR**, REAL(SP),intent(in) **VALUE**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**, \*)

**Parameters:**

**VECTOR** A pointer to the vector to set  
**VALUE** The value to set the vector to  
**ERR** The error code  
**ERROR** The error string

Definition at line 4042 of file matrix\_vector.f90.

## 7.137 MATRIX\_VECTOR::VECTOR\_DATA\_GET Interface Reference

### Public Member Functions

- subroutine [VECTOR\\_DATA\\_GET\\_INTG](#) (VECTOR, DATA, ERR, ERROR,\*)
- subroutine [VECTOR\\_DATA\\_GET\\_SP](#) (VECTOR, DATA, ERR, ERROR,\*)
- subroutine [VECTOR\\_DATA\\_GET\\_DP](#) (VECTOR, DATA, ERR, ERROR,\*)
- subroutine [VECTOR\\_DATA\\_GET\\_L](#) (VECTOR, DATA, ERR, ERROR,\*)

#### 7.137.1 Detailed Description

Definition at line 244 of file matrix\_vector.f90.

#### 7.137.2 Member Function Documentation

**7.137.2.1 subroutine MATRIX\_VECTOR::VECTOR\_DATA\_GET::VECTOR\_DATA\_GET\_DP** (TYPE(VECTOR\_TYPE),pointer *VECTOR*, REAL(DP),dimension(:),pointer *DATA*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

##### Parameters:

**VECTOR** A pointer to the vector

**DATA** On return a pointer to the vector data

**ERR** The error code

**ERROR** The error string

Definition at line 4339 of file matrix\_vector.f90.

**7.137.2.2 subroutine MATRIX\_VECTOR::VECTOR\_DATA\_GET::VECTOR\_DATA\_GET\_INTG** (TYPE(VECTOR\_TYPE),pointer *VECTOR*, INTEGER(INTG),dimension(:),pointer *DATA*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

##### Parameters:

**VECTOR** A pointer to the vector

**DATA** On return a pointer to the vector data

**ERR** The error code

**ERROR** The error string

Definition at line 4249 of file matrix\_vector.f90.

7.137.2.3 subroutine MATRIX\_VECTOR::VECTOR\_DATA\_GET::VECTOR\_DATA\_GET\_L  
(TYPE(VECTOR\_TYPE),pointer **VECTOR**, LOGICAL,dimension(:),pointer **DATA**,  
INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**,  
\*)

**Parameters:**

**VECTOR** A pointer to the vector  
**DATA** On return a pointer to the vector data  
**ERR** The error code  
**ERROR** The error string

Definition at line 4384 of file matrix\_vector.f90.

7.137.2.4 subroutine MATRIX\_VECTOR::VECTOR\_DATA\_GET::VECTOR\_DATA\_GET\_SP  
(TYPE(VECTOR\_TYPE),pointer **VECTOR**, REAL(SP),dimension(:),pointer **DATA**,  
INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**,  
\*)

**Parameters:**

**VECTOR** A pointer to the vector  
**DATA** On return a pointer to the vector data  
**ERR** The error code  
**ERROR** The error string

Definition at line 4294 of file matrix\_vector.f90.

## 7.138 MATRIX\_VECTOR::VECTOR\_VALUES\_GET Interface Reference

### Public Member Functions

- subroutine [VECTOR\\_VALUES\\_GET\\_INTG](#) (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
- subroutine [VECTOR\\_VALUES\\_GET\\_INTG1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
- subroutine [VECTOR\\_VALUES\\_GET\\_SP](#) (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
- subroutine [VECTOR\\_VALUES\\_GET\\_SP1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
- subroutine [VECTOR\\_VALUES\\_GET\\_DP](#) (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
- subroutine [VECTOR\\_VALUES\\_GET\\_DP1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
- subroutine [VECTOR\\_VALUES\\_GET\\_L](#) (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
- subroutine [VECTOR\\_VALUES\\_GET\\_L1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,\*)

#### 7.138.1 Detailed Description

Definition at line 251 of file matrix\_vector.f90.

#### 7.138.2 Member Function Documentation

**7.138.2.1 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_-  
GET::VECTOR\_VALUES\_GET\_DP (TYPE(VECTOR\_TYPE),pointer  
VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES,  
REAL(DP),dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

##### Parameters:

**VECTOR** A pointer to the vector  
**INDICES** INDICES(i). The i'th index to get  
**VALUES** VALUES(i). On return the i'th value to get  
**ERR** The error code  
**ERROR** The error string

Definition at line 4853 of file matrix\_vector.f90.

**7.138.2.2 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_GET::VECTOR\_VALUES\_-  
GET\_DP1 (TYPE(VECTOR\_TYPE),pointer VECTOR, INTEGER(INTG),intent(in)  
INDEX, REAL(DP),intent(out) VALUE, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

##### Parameters:

**VECTOR** A pointer to the vector  
**INDEX** The index of the vector to get  
**VALUE** On return the value of the vector at the specified index  
**ERR** The error code  
**ERROR** The error string

Definition at line 4911 of file matrix\_vector.f90.

---

**7.138.2.3 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_GET::VECTOR\_VALUES\_GET\_INTG (TYPE(VECTOR\_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, INTEGER(INTG),dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

**Parameters:**

**VECTOR** A pointer to the vector  
**INDICES** INDICES(i). The i'th index to get  
**VALUES** VALUES(i). On return the i'th value to get  
**ERR** The error code  
**ERROR** The error string

Definition at line 4643 of file matrix\_vector.f90.

**7.138.2.4 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_GET::VECTOR\_VALUES\_GET\_INTG1 (TYPE(VECTOR\_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, INTEGER(INTG),intent(out) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

**Parameters:**

**VECTOR** A pointer to the vector  
**INDEX** The index of the vector to get  
**VALUE** On return the value of the vector at the specified index  
**ERR** The error code  
**ERROR** The error string

Definition at line 4701 of file matrix\_vector.f90.

**7.138.2.5 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_GET::VECTOR\_VALUES\_GET\_L (TYPE(VECTOR\_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, LOGICAL,dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

**Parameters:**

**VECTOR** A pointer to the vector  
**INDICES** INDICES(i). The i'th index to get  
**VALUES** VALUES(i). On return the i'th value to get  
**ERR** The error code  
**ERROR** The error string

Definition at line 4958 of file matrix\_vector.f90.

**7.138.2.6 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_GET::VECTOR\_VALUES\_GET\_L1 (TYPE(VECTOR\_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, LOGICAL,intent(out) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

**Parameters:**

**VECTOR** A pointer to the vector

**INDEX** The index of the vector to get

**VALUE** On return the value of the vector at the specified index

**ERR** The error code

**ERROR** The error string

Definition at line 5016 of file matrix\_vector.f90.

**7.138.2.7 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_GET::VECTOR\_VALUES\_GET\_SP (TYPE(VECTOR\_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, REAL(SP),dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

**Parameters:**

**VECTOR** A pointer to the vector

**INDICES** INDICES(i). The i'th index to get

**VALUES** VALUES(i). On return the i'th value to get

**ERR** The error code

**ERROR** The error string

Definition at line 4748 of file matrix\_vector.f90.

**7.138.2.8 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_GET::VECTOR\_VALUES\_GET\_SP1 (TYPE(VECTOR\_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, REAL(SP),intent(out) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

**Parameters:**

**VECTOR** A pointer to the vector

**INDEX** The index of the vector to get

**VALUE** On return the value of the vector at the specified index

**ERR** The error code

**ERROR** The error string

Definition at line 4806 of file matrix\_vector.f90.

## 7.139 MATRIX\_VECTOR::VECTOR\_VALUES\_SET Interface Reference

### Public Member Functions

- subroutine `VECTOR_VALUES_SET_INTG` (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
- subroutine `VECTOR_VALUES_SET_INTG1` (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
- subroutine `VECTOR_VALUES_SET_SP` (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
- subroutine `VECTOR_VALUES_SET_SP1` (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
- subroutine `VECTOR_VALUES_SET_DP` (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
- subroutine `VECTOR_VALUES_SET_DP1` (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
- subroutine `VECTOR_VALUES_SET_L` (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
- subroutine `VECTOR_VALUES_SET_L1` (VECTOR, INDEX, VALUE, ERR, ERROR,\*)

#### 7.139.1 Detailed Description

Definition at line 262 of file matrix\_vector.f90.

#### 7.139.2 Member Function Documentation

**7.139.2.1 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_SET::VECTOR\_VALUES\_SET\_DP** (TYPE(VECTOR\_TYPE),pointer *VECTOR*, INTEGER(INTG),dimension(:),intent(in) *INDICES*, REAL(DP),dimension(:),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:**

**VECTOR** A pointer to the vector  
**INDICES** INDICES(i). The i'th index to set  
**VALUES** VALUES(i). The i'th value to set  
**ERR** The error code  
**ERROR** The error string

Definition at line 5273 of file matrix\_vector.f90.

**7.139.2.2 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_SET::VECTOR\_VALUES\_SET\_DP1** (TYPE(VECTOR\_TYPE),pointer *VECTOR*, INTEGER(INTG),intent(in) *INDEX*, REAL(DP),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:**

**VECTOR** A pointer to the vector  
**INDEX** The index to set  
**VALUE** The value to set at the specified index  
**ERR** The error code  
**ERROR** The error string

Definition at line 5331 of file matrix\_vector.f90.

**7.139.2.3 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_SET::VECTOR\_VALUES\_SET\_INTG (TYPE(VECTOR\_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, INTEGER(INTG),dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

**Parameters:**

**VECTOR** A pointer to the vector  
**INDICES** INDICES(i). The i'th index to set  
**VALUES** VALUES(i). The i'th value to set  
**ERR** The error code  
**ERROR** The error string

Definition at line 5063 of file matrix\_vector.f90.

**7.139.2.4 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_SET::VECTOR\_VALUES\_SET\_INTG1 (TYPE(VECTOR\_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, INTEGER(INTG),intent(in) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

**Parameters:**

**VECTOR** A pointer to the vector  
**INDEX** The index to set  
**VALUE** The value to set at the specified index  
**ERR** The error code  
**ERROR** The error string

Definition at line 5121 of file matrix\_vector.f90.

**7.139.2.5 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_SET::VECTOR\_VALUES\_SET\_L (TYPE(VECTOR\_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, LOGICAL,dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*)**

**Parameters:**

**VECTOR** A pointer to the vector  
**INDICES** INDICES(i). The i'th index to set  
**VALUES** VALUES(i). The i'th value to set  
**ERR** The error code  
**ERROR** The error string

Definition at line 5378 of file matrix\_vector.f90.

---

**7.139.2.6 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_SET::VECTOR\_VALUES\_SET\_L1** (TYPE(VECTOR\_TYPE),pointer *VECTOR*, INTEGER(INTG),intent(in) *INDEX*, LOGICAL,intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:**

***VECTOR*** A pointer to the vector  
***INDEX*** The index to set  
***VALUE*** The value to set at the specified index  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 5436 of file matrix\_vector.f90.

**7.139.2.7 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_SET::VECTOR\_VALUES\_SET\_SP** (TYPE(VECTOR\_TYPE),pointer *VECTOR*, INTEGER(INTG),dimension(:),intent(in) *INDICES*, REAL(SP),dimension(:),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:**

***VECTOR*** A pointer to the vector  
***INDICES*** INDICES(i). The i'th index to set  
***VALUES*** VALUES(i). The i'th value to set  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 5168 of file matrix\_vector.f90.

**7.139.2.8 subroutine MATRIX\_VECTOR::VECTOR\_VALUES\_SET::VECTOR\_VALUES\_SET\_SP1** (TYPE(VECTOR\_TYPE),pointer *VECTOR*, INTEGER(INTG),intent(in) *INDEX*, REAL(SP),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*)

**Parameters:**

***VECTOR*** A pointer to the vector  
***INDEX*** The index to set  
***VALUE*** The value to set at the specified index  
***ERR*** The error code  
***ERROR*** The error string

Definition at line 5226 of file matrix\_vector.f90.

## **7.140 MESH\_ROUTINES::MESH\_NUMBER\_OF\_COMPONENTS\_SET Interface Reference**

Sets/changes the number of mesh components for a mesh.

### **Private Member Functions**

- subroutine `MESH_NUMBER_OF_COMPONENTS_SET_NUMBER` (`USER_NUMBER`, `REGION`, `NUMBER_OF_COMPONENTS`, `ERR`, `ERROR,*`)
- subroutine `MESH_NUMBER_OF_COMPONENTS_SET_PTR` (`MESH`, `NUMBER_OF_COMPONENTS`, `ERR`, `ERROR,*`)

### **7.140.1 Detailed Description**

Sets/changes the number of mesh components for a mesh.

Definition at line 85 of file mesh\_routines.f90.

### **7.140.2 Member Function Documentation**

**7.140.2.1 subroutine MESH\_ROUTINES::MESH\_NUMBER\_OF\_COMPONENTS\_SET::MESH\_NUMBER\_OF\_COMPONENTS\_SET\_NUMBER**  
(`INTEGER(INTG),intent(in) USER_NUMBER`, `TYPE(REGION_TYPE),pointer REGION`, `INTEGER(INTG),intent(in) NUMBER_OF_COMPONENTS`,  
`INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR`,  
\*) [private]

#### **Parameters:**

`USER_NUMBER` The user number of the mesh to set the number of components for  
`REGION` A pointer to the region containing the mesh

`NUMBER_OF_COMPONENTS` The number of components to set for the mesh

`ERR` The error code

`ERROR` The error string

Definition at line 4639 of file mesh\_routines.f90.

**7.140.2.2 subroutine MESH\_ROUTINES::MESH\_NUMBER\_OF\_COMPONENTS\_SET::MESH\_NUMBER\_OF\_COMPONENTS\_SET\_PTR**  
(`TYPE(MESH_TYPE),pointer MESH`, `INTEGER(INTG),intent(in) NUMBER_OF_COMPONENTS`, `INTEGER(INTG),intent(out) ERR`,  
`TYPE(VARYING_STRING),intent(out) ERROR`, \*) [private]

#### **Parameters:**

`MESH` A pointer to the mesh to set the number of components for

`NUMBER_OF_COMPONENTS` The number of components to set.

`ERR` The error code

`ERROR` The error string

Definition at line 4677 of file mesh\_routines.f90.

## 7.141 MESH\_ROUTINES::MESH\_NUMBER\_OF\_ELEMENTS\_- SET Interface Reference

Sets/changes the number of elements for a mesh.

### Private Member Functions

- subroutine [MESH\\_NUMBER\\_OF\\_ELEMENTS\\_SET\\_NUMBER](#) (*USER\_NUMBER*, *REGION*, *NUMBER\_OF\_ELEMENTS*, *ERR*, *ERROR*,\*)
- subroutine [MESH\\_NUMBER\\_OF\\_ELEMENTS\\_SET\\_PTR](#) (*MESH*, *NUMBER\_OF\_ELEMENTS*, *ERR*, *ERROR*,\*)

### 7.141.1 Detailed Description

Sets/changes the number of elements for a mesh.

Definition at line 91 of file mesh\_routines.f90.

### 7.141.2 Member Function Documentation

**7.141.2.1 subroutine MESH\_ROUTINES::MESH\_NUMBER\_OF\_ELEMENTS\_-  
SET::MESH\_NUMBER\_OF\_ELEMENTS\_SET\_NUMBER**  
*(INTEGER(INTG),intent(in) USER\_NUMBER, TYPE(REGION\_TYPE),pointer  
 REGION, INTEGER(INTG),intent(in) NUMBER\_OF\_ELEMENTS,  
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
 \*) [private]*

#### Parameters:

**USER\_NUMBER** The user number of the mesh to set the number of elements for  
**REGION** A pointer to the region containing the mesh  
**NUMBER\_OF\_ELEMENTS** The number of elements to set  
**ERR** The error code  
**ERROR** The error string

Definition at line 4783 of file mesh\_routines.f90.

**7.141.2.2 subroutine MESH\_ROUTINES::MESH\_NUMBER\_OF\_ELEMENTS\_-  
SET::MESH\_NUMBER\_OF\_ELEMENTS\_SET\_PTR** (*TYPE(MESH\_TYPE),pointer  
 MESH, INTEGER(INTG),intent(in) NUMBER\_OF\_ELEMENTS,  
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR,  
 \*) [private]*)

#### Parameters:

**MESH** A pointer to the mesh to set the number of elements for  
**NUMBER\_OF\_ELEMENTS** The number of elements to set  
**ERR** The error code  
**ERROR** The error string

Definition at line 4821 of file mesh\_routines.f90.

## 7.142 PROBLEM\_ROUTINES::PROBLEM\_DESTROY Interface Reference

### Private Member Functions

- subroutine [PROBLEM\\_DESTROY\\_NUMBER](#) (USER\_NUMBER, ERR, ERROR,\*)
- subroutine [PROBLEM\\_DESTROY\\_PTR](#) (PROBLEM, ERR, ERROR,\*)

#### 7.142.1 Detailed Description

Definition at line 98 of file problem\_routines.f90.

#### 7.142.2 Member Function Documentation

**7.142.2.1 subroutine PROBLEM\_ROUTINES::PROBLEM\_DESTROY::PROBLEM\_DESTROY\_NUMBER (INTEGER(INTG),intent(in) USER\_NUMBER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

##### Parameters:

**USER\_NUMBER** The user number of the problem to destroy

**ERR** The error code

**ERROR** The error string

Definition at line 261 of file problem\_routines.f90.

**7.142.2.2 subroutine PROBLEM\_ROUTINES::PROBLEM\_DESTROY::PROBLEM\_DESTROY\_PTR (TYPE(PROBLEM\_TYPE),pointer PROBLEM, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, \*) [private]**

##### Parameters:

**PROBLEM** A pointer to the problem to destroy

**ERR** The error code

**ERROR** The error string

Definition at line 312 of file problem\_routines.f90.

## 7.143 PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_- GET Interface Reference

### Private Member Functions

- subroutine **PROBLEM\_SPECIFICATION\_GET\_NUMBER** (USER\_NUMBER, PROBLEM\_CLASS, PROBLEM\_EQUATION\_TYPE, PROBLEM\_SUBTYPE, ERR, ERROR,\*)
- subroutine **PROBLEM\_SPECIFICATION\_GET\_PTR** (PROBLEM, PROBLEM\_CLASS, PROBLEM\_EQUATION\_TYPE, PROBLEM\_SUBTYPE, ERR, ERROR,\*)

#### 7.143.1 Detailed Description

Definition at line 103 of file problem\_routines.f90.

#### 7.143.2 Member Function Documentation

**7.143.2.1 subroutine PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_-  
GET::PROBLEM\_SPECIFICATION\_GET\_NUMBER** (INTEGER(INTG),intent(in)  
*USER\_NUMBER*, INTEGER(INTG),intent(out) *PROBLEM\_CLASS*,  
INTEGER(INTG),intent(out) *PROBLEM\_EQUATION\_TYPE*,  
INTEGER(INTG),intent(out) *PROBLEM\_SUBTYPE*, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

##### Parameters:

**USER\_NUMBER** The user number of the problem to set the specification for.

**PROBLEM\_CLASS** The problem class to get.

**PROBLEM\_EQUATION\_TYPE** The problem equation to get.

**PROBLEM\_SUBTYPE** The problem subtype.

**ERR** The error code

**ERROR** The error string

Definition at line 1346 of file problem\_routines.f90.

**7.143.2.2 subroutine PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_-  
GET::PROBLEM\_SPECIFICATION\_GET\_PTR** (TYPE(PROBLEM\_TYPE),pointer  
*PROBLEM*, INTEGER(INTG),intent(out) *PROBLEM\_CLASS*,  
INTEGER(INTG),intent(out) *PROBLEM\_EQUATION\_TYPE*,  
INTEGER(INTG),intent(out) *PROBLEM\_SUBTYPE*, INTEGER(INTG),intent(out)  
*ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

##### Parameters:

**PROBLEM** A pointer to the problem to set the specification for.

**PROBLEM\_CLASS** The problem class to set.

**PROBLEM\_EQUATION\_TYPE** The problem equation type to set.

**PROBLEM\_SUBTYPE** The problem subtype to set.

**ERR** The error code

***ERROR*** The error string

Definition at line 1375 of file problem\_routines.f90.

## 7.144 PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_SET Interface Reference

### Private Member Functions

- subroutine **PROBLEM\_SPECIFICATION\_SET\_NUMBER** (*USER\_NUMBER*, *PROBLEM\_CLASS*, *PROBLEM\_EQUATION\_TYPE*, *PROBLEM\_SUBTYPE*, *ERR*, *ERROR*,\*)
- subroutine **PROBLEM\_SPECIFICATION\_SET\_PTR** (*PROBLEM*, *PROBLEM\_CLASS*, *PROBLEM\_EQUATION\_TYPE*, *PROBLEM\_SUBTYPE*, *ERR*, *ERROR*,\*)

#### 7.144.1 Detailed Description

Definition at line 108 of file problem\_routines.f90.

#### 7.144.2 Member Function Documentation

**7.144.2.1 subroutine PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_SET::PROBLEM\_SPECIFICATION\_SET\_NUMBER** (INTEGER(INTG),intent(in) *USER\_NUMBER*, INTEGER(INTG),intent(in) *PROBLEM\_CLASS*, INTEGER(INTG),intent(in) *PROBLEM\_EQUATION\_TYPE*, INTEGER(INTG),intent(in) *PROBLEM\_SUBTYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

##### Parameters:

***USER\_NUMBER*** The user number of the problem to set the specification for.

***PROBLEM\_CLASS*** The problem class to set.

***PROBLEM\_EQUATION\_TYPE*** The problem equation to set.

***PROBLEM\_SUBTYPE*** The problem subtype.

***ERR*** The error code

***ERROR*** The error string

Definition at line 1430 of file problem\_routines.f90.

**7.144.2.2 subroutine PROBLEM\_ROUTINES::PROBLEM\_SPECIFICATION\_SET::PROBLEM\_SPECIFICATION\_SET\_PTR** (TYPE(*PROBLEM\_TYPE*),pointer *PROBLEM*, INTEGER(INTG),intent(in) *PROBLEM\_CLASS*, INTEGER(INTG),intent(in) *PROBLEM\_EQUATION\_TYPE*, INTEGER(INTG),intent(in) *PROBLEM\_SUBTYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

##### Parameters:

***PROBLEM*** A pointer to the problem to set the specification for.

***PROBLEM\_CLASS*** The problem class to set.

***PROBLEM\_EQUATION\_TYPE*** The problem equation type to set.

***PROBLEM\_SUBTYPE*** The problem subtype to set.

***ERR*** The error code

***ERROR*** The error string

Definition at line 1459 of file problem\_routines.f90.

## 7.145 REGION\_ROUTINES::REGION\_COORDINATE\_- SYSTEM\_SET Interface Reference

### Private Member Functions

- subroutine [REGION\\_COORDINATE\\_SYSTEM\\_SET\\_NUMBER](#) (USER\_NUMBER,  
COORDINATE\_SYSTEM, ERR, ERROR,\*)
- subroutine [REGION\\_COORDINATE\\_SYSTEM\\_SET\\_PTR](#) (REGION, COORDINATE\_SYSTEM,  
ERR, ERROR,\*)

#### 7.145.1 Detailed Description

Definition at line 71 of file region\_routines.f90.

#### 7.145.2 Member Function Documentation

7.145.2.1 subroutine **REGION\_ROUTINES::REGION\_COORDINATE\_-  
SYSTEM\_SET:::REGION\_COORDINATE\_SYSTEM\_SET\_NUMBER**  
(INTEGER(INTG),intent(in) *USER\_NUMBER*, TYPE(COORDINATE\_SYSTEM\_-  
TYPE),pointer *COORDINATE\_SYSTEM*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 135 of file region\_routines.f90.

7.145.2.2 subroutine **REGION\_ROUTINES::REGION\_COORDINATE\_-  
SYSTEM\_SET:::REGION\_COORDINATE\_SYSTEM\_SET\_PTR**  
(TYPE(REGION\_TYPE),pointer *REGION*, TYPE(COORDINATE\_SYSTEM\_-  
TYPE),pointer *COORDINATE\_SYSTEM*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 165 of file region\_routines.f90.

## 7.146 REGION\_ROUTINES::REGION\_LABEL\_SET Interface Reference

### Private Member Functions

- subroutine [REGION\\_LABEL\\_SET\\_NUMBER](#) (USER\_NUMBER, LABEL, ERR, ERROR,\*)
- subroutine [REGION\\_LABEL\\_SET\\_PTR](#) (REGION, LABEL, ERR, ERROR,\*)

#### 7.146.1 Detailed Description

Definition at line 77 of file region\_routines.f90.

#### 7.146.2 Member Function Documentation

**7.146.2.1 subroutine REGION\_ROUTINES::REGION\_LABEL\_SET::REGION\_LABEL\_SET\_NUMBER (INTEGER(INTG),intent(in) *USER\_NUMBER*, CHARACTER(LEN=\*),intent(in) *LABEL*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Definition at line 462 of file region\_routines.f90.

**7.146.2.2 subroutine REGION\_ROUTINES::REGION\_LABEL\_SET::REGION\_LABEL\_SET\_PTR (TYPE(REGION\_TYPE),pointer *REGION*, CHARACTER(LEN=\*),intent(in) *LABEL*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]**

Definition at line 492 of file region\_routines.f90.

## 7.147 SORTING::BUBBLE\_SORT Interface Reference

### Private Member Functions

- subroutine [BUBBLE\\_SORT\\_INTG](#) (A, ERR, ERROR,\*)
- subroutine [BUBBLE\\_SORT\\_SP](#) (A, ERR, ERROR,\*)
- subroutine [BUBBLE\\_SORT\\_DP](#) (A, ERR, ERROR,\*)

#### 7.147.1 Detailed Description

Definition at line 61 of file sorting.f90.

#### 7.147.2 Member Function Documentation

**7.147.2.1 subroutine [SORTING::BUBBLE\\_SORT::BUBBLE\\_SORT\\_DP](#)**  
(REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 185 of file sorting.f90.

**7.147.2.2 subroutine [SORTING::BUBBLE\\_SORT::BUBBLE\\_SORT\\_INTG](#)**  
(INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 96 of file sorting.f90.

**7.147.2.3 subroutine [SORTING::BUBBLE\\_SORT::BUBBLE\\_SORT\\_SP](#)**  
(REAL(SP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 140 of file sorting.f90.

## 7.148 SORTING::HEAP\_SORT Interface Reference

### Private Member Functions

- subroutine [HEAP\\_SORT\\_INTG](#) (A, ERR, ERROR,\*)
- subroutine [HEAP\\_SORT\\_SP](#) (A, ERR, ERROR,\*)
- subroutine [HEAP\\_SORT\\_DP](#) (A, ERR, ERROR,\*)

#### 7.148.1 Detailed Description

Definition at line 67 of file sorting.f90.

#### 7.148.2 Member Function Documentation

**7.148.2.1 subroutine SORTING::HEAP\_SORT::HEAP\_SORT\_DP**  
(REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 362 of file sorting.f90.

**7.148.2.2 subroutine SORTING::HEAP\_SORT::HEAP\_SORT\_INTG**  
(INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 239 of file sorting.f90.

**7.148.2.3 subroutine SORTING::HEAP\_SORT::HEAP\_SORT\_SP**  
(REAL(SP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 300 of file sorting.f90.

## 7.149 SORTING::SHELL\_SORT Interface Reference

### Private Member Functions

- subroutine [SHELL\\_SORT\\_INTG](#) (A, ERR, ERROR,\*)
- subroutine [SHELL\\_SORT\\_SP](#) (A, ERR, ERROR,\*)
- subroutine [SHELL\\_SORT\\_DP](#) (A, ERR, ERROR,\*)

#### 7.149.1 Detailed Description

Definition at line 73 of file sorting.f90.

#### 7.149.2 Member Function Documentation

**7.149.2.1 subroutine [SORTING::SHELL\\_SORT::SHELL\\_SORT\\_DP](#)**  
(REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 524 of file sorting.f90.

**7.149.2.2 subroutine [SORTING::SHELL\\_SORT::SHELL\\_SORT\\_INTG](#)**  
(INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 433 of file sorting.f90.

**7.149.2.3 subroutine [SORTING::SHELL\\_SORT::SHELL\\_SORT\\_SP](#)**  
(REAL(SP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*, \*) [private]

Definition at line 478 of file sorting.f90.

## 7.150 STRINGS::CHARACTER\_TO\_LOWERCASE Interface Reference

Returns a character string which is the lowercase equivalent of the supplied string.

### Private Member Functions

- function [CHARACTER\\_TO\\_LOWERCASE\\_C](#) (STRING)
- function [CHARACTER\\_TO\\_LOWERCASE\\_VS](#) (STRING)

#### 7.150.1 Detailed Description

Returns a character string which is the lowercase equivalent of the supplied string.

Definition at line 126 of file strings.f90.

#### 7.150.2 Member Function Documentation

##### 7.150.2.1 function STRINGS::CHARACTER\_TO\_LOWERCASE::CHARACTER\_TO\_LOWERCASE\_C (CHARACTER(LEN=\*)*intent(in)* STRING) [private]

###### Parameters:

*STRING* The string to convert to lowercase

Definition at line 1609 of file strings.f90.

##### 7.150.2.2 function STRINGS::CHARACTER\_TO\_LOWERCASE::CHARACTER\_TO\_LOWERCASE\_VS (TYPE(VARYING\_STRING)*intent(in)* STRING) [private]

###### Parameters:

*STRING* The string to convert

Definition at line 1634 of file strings.f90.

## 7.151 STRINGS::CHARACTER\_TO\_UPPERCASE Interface Reference

Returns a character string which is the uppercase equivalent of the supplied string.

### Private Member Functions

- function [CHARACTER\\_TO\\_UPPERCASE\\_C](#) (STRING)
- function [CHARACTER\\_TO\\_UPPERCASE\\_VS](#) (STRING)

#### 7.151.1 Detailed Description

Returns a character string which is the uppercase equivalent of the supplied string.

Definition at line 138 of file strings.f90.

#### 7.151.2 Member Function Documentation

**7.151.2.1 function STRINGS::CHARACTER\_TO\_UPPERCASE::CHARACTER\_TO\_UPPERCASE\_C (CHARACTER(LEN=\*)*intent(in)* STRING)**  
[private]

##### Parameters:

**STRING** The string to convert

Definition at line 1709 of file strings.f90.

**7.151.2.2 function STRINGS::CHARACTER\_TO\_UPPERCASE::CHARACTER\_TO\_UPPERCASE\_VS (TYPE(VARYING\_STRING)*intent(in)* STRING)**  
[private]

##### Parameters:

**STRING** The string to convert

Definition at line 1734 of file strings.f90.

## 7.152 STRINGS::IS\_ABBREVIATION Interface Reference

Returns .TRUE. if a supplied string is a valid abbreviation of a second supplied string.

### Private Member Functions

- LOGICAL [IS\\_ABBREVIATION\\_C\\_C](#) (SHORT, LONG, MIN\_NUM\_CHARACTERS)
- LOGICAL [IS\\_ABBREVIATION\\_C\\_VS](#) (SHORT, LONG, MIN\_NUM\_CHARACTERS)
- LOGICAL [IS\\_ABBREVIATION\\_VS\\_C](#) (SHORT, LONG, MIN\_NUM\_CHARACTERS)
- LOGICAL [IS\\_ABBREVIATION\\_VS\\_VS](#) (SHORT, LONG, MIN\_NUM\_CHARACTERS)

### 7.152.1 Detailed Description

Returns .TRUE. if a supplied string is a valid abbreviation of a second supplied string.

Definition at line 62 of file strings.f90.

### 7.152.2 Member Function Documentation

#### 7.152.2.1 LOGICAL STRINGS::IS\_ABBREVIATION::IS\_ABBREVIATION\_C\_C (CHARACTER(LEN=\*),intent(in) SHORT, CHARACTER(LEN=\*),intent(in) LONG, INTEGER(INTG),intent(in) MIN\_NUM\_CHARACTERS) [private]

##### Parameters:

**SHORT** The short form of the string

**LONG** The long form of the string

**MIN\_NUM\_CHARACTERS** The minimum number of characters to match

Definition at line 160 of file strings.f90.

#### 7.152.2.2 LOGICAL STRINGS::IS\_ABBREVIATION::IS\_ABBREVIATION\_C\_VS (CHARACTER(LEN=\*),intent(in) SHORT, TYPE(VARYING\_STRING),intent(in) LONG, INTEGER(INTG),intent(in) MIN\_NUM\_CHARACTERS) [private]

##### Parameters:

**SHORT** The short form of the string

**LONG** The long form of the string

**MIN\_NUM\_CHARACTERS** The minimum number of characters to match

Definition at line 192 of file strings.f90.

#### 7.152.2.3 LOGICAL STRINGS::IS\_ABBREVIATION::IS\_ABBREVIATION\_VS\_C (TYPE(VARYING\_STRING),intent(in) SHORT, CHARACTER(LEN=\*),intent(in) LONG, INTEGER(INTG),intent(in) MIN\_NUM\_CHARACTERS) [private]

##### Parameters:

**SHORT** The short form of the string

**LONG** The long form of the string

**MIN\_NUM\_CHARACTERS** The minimum number of characters to match

Definition at line 224 of file strings.f90.

**7.152.2.4 LOGICAL\_STRINGS::IS\_ABBREVIATION::IS\_ABBREVIATION\_VS\_VS**  
(**TYPE(VARYING\_STRING),intent(in)**) **SHORT**, **TYPE(VARYING\_STRING),intent(in)**  
**LONG**, **INTEGER(INTG),intent(in)** **MIN\_NUM\_CHARACTERS** [private]

**Parameters:**

**SHORT** The short form of the string

**LONG** The long form of the string

**MIN\_NUM\_CHARACTERS** The minimum number of characters to match

Definition at line 256 of file strings.f90.

## 7.153 STRINGS::LIST\_TO\_CHARACTER Interface Reference

Converts a list to its equivalent character string representation.

### Private Member Functions

- CHARACTER(LEN=MAXSTRLEN) [LIST\\_TO\\_CHARACTER\\_C](#) (NUMBER\_IN\_LIST, LIST, FORMAT, ERR, ERROR, LIST\_LENGTHS)
- CHARACTER(LEN=MAXSTRLEN) [LIST\\_TO\\_CHARACTER\\_INTG](#) (NUMBER\_IN\_LIST, LIST, FORMAT, ERR, ERROR)
- CHARACTER(LEN=MAXSTRLEN) [LIST\\_TO\\_CHARACTER\\_LINTG](#) (NUMBER\_IN\_LIST, LIST, FORMAT, ERR, ERROR)
- CHARACTER(LEN=MAXSTRLEN) [LIST\\_TO\\_CHARACTER\\_L](#) (NUMBER\_IN\_LIST, LIST, FORMAT, ERR, ERROR)
- CHARACTER(LEN=MAXSTRLEN) [LIST\\_TO\\_CHARACTER\\_SP](#) (NUMBER\_IN\_LIST, LIST, FORMAT, ERR, ERROR)
- CHARACTER(LEN=MAXSTRLEN) [LIST\\_TO\\_CHARACTER\\_DP](#) (NUMBER\_IN\_LIST, LIST, FORMAT, ERR, ERROR)

### 7.153.1 Detailed Description

Converts a list to its equivalent character string representation.

Definition at line 70 of file strings.f90.

### 7.153.2 Member Function Documentation

**7.153.2.1 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST\_TO\_CHARACTER::LIST\_TO\_CHARACTER\_C (INTEGER(INTG),intent(in) NUMBER\_IN\_LIST, CHARACTER(LEN=\*),dimension(number\_in\_list),intent(in) LIST, CHARACTER(LEN=\*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR, INTEGER(INTG),dimension(number\_in\_list),intent(in),optional LIST\_LENGTHS) [private]**

#### Parameters:

**NUMBER\_IN\_LIST** The number of items in the list

**LIST** LIST(i). The i'th item in the list

**FORMAT** The format to use. Ignored for character lists.

**ERR** The error code

**ERROR** The error string

**LIST\_LENGTHS** LIST\_LENGTHS(i). Optional, The length of the i'th list item.

Definition at line 387 of file strings.f90.

---

**7.153.2.2 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST\_TO\_-  
CHARACTER::LIST\_TO\_CHARACTER\_DP (INTEGER(INTG),intent(in)  
NUMBER\_IN\_LIST, REAL(DP),dimension(number\_in\_list),intent(in) LIST,  
CHARACTER(LEN=\*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

**Parameters:**

**NUMBER\_IN\_LIST** The number of items in the list  
**LIST** LIST(i). The i'th item in the list  
**FORMAT** The format to use for the conversion  
**ERR** The error code  
**ERROR** The error string

Definition at line 668 of file strings.f90.

**7.153.2.3 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST\_TO\_CHARACTER::LIST\_-  
TO\_CHARACTER\_INTG (INTEGER(INTG),intent(in) NUMBER\_IN\_LIST,  
INTEGER(INTG),dimension(number\_in\_list),intent(in) LIST,  
CHARACTER(LEN=\*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

**Parameters:**

**NUMBER\_IN\_LIST** The number of items in the list  
**LIST** LIST(i). The i'th item in the list  
**FORMAT** The format to use for the conversion  
**ERR** The error code  
**ERROR** The error string

Definition at line 458 of file strings.f90.

**7.153.2.4 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST\_TO\_-  
CHARACTER::LIST\_TO\_CHARACTER\_L (INTEGER(INTG),intent(in)  
NUMBER\_IN\_LIST, LOGICAL,dimension(number\_in\_list),intent(in) LIST,  
CHARACTER(LEN=\*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

**Parameters:**

**NUMBER\_IN\_LIST** The number of items in the list  
**LIST** LIST(i). The i'th item in the list  
**FORMAT** The format to use. Ignored for logical lists.  
**ERR** The error code  
**ERROR** The error string

Definition at line 564 of file strings.f90.

**7.153.2.5 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST\_TO\_CHARACTER::LIST\_TO\_CHARACTER\_LINTG (INTEGER(INTG),intent(in) *NUMBER\_IN\_LIST*,  
INTEGER(LINTG),dimension(*number\_in\_list*),intent(in) *LIST*,  
CHARACTER(LEN=\*),intent(in) *FORMAT*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]**

**Parameters:**

***NUMBER\_IN\_LIST*** The number of items in the list

***LIST*** LIST(i). The i'th item in the list

***FORMAT*** The format to use for the conversion

***ERR*** The error code

***ERROR*** The error string

Definition at line 511 of file strings.f90.

**7.153.2.6 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST\_TO\_CHARACTER::LIST\_TO\_CHARACTER\_SP (INTEGER(INTG),intent(in) *NUMBER\_IN\_LIST*, REAL(SP),dimension(*number\_in\_list*),intent(in) *LIST*,  
CHARACTER(LEN=\*),intent(in) *FORMAT*, INTEGER(INTG),intent(out) *ERR*,  
TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]**

**Parameters:**

***NUMBER\_IN\_LIST*** The number of items in the list

***LIST*** LIST(i). The i'th item in the list

***FORMAT*** The format to use for the conversion

***ERR*** The error code

***ERROR*** The error string

Definition at line 616 of file strings.f90.

## 7.154 STRINGS::NUMBER\_TO\_CHARACTER Interface Reference

Converts a number to its equivalent character string representation.

### Private Member Functions

- CHARACTER(LEN=MAXSTRLEN) [NUMBER\\_TO\\_CHARACTER\\_INTG](#) (NUMBER, FORMAT, ERR, ERROR)
- CHARACTER(LEN=MAXSTRLEN) [NUMBER\\_TO\\_CHARACTER\\_LINTG](#) (NUMBER, FORMAT, ERR, ERROR)
- CHARACTER(LEN=MAXSTRLEN) [NUMBER\\_TO\\_CHARACTER\\_SP](#) (NUMBER, FORMAT, ERR, ERROR)
- CHARACTER(LEN=MAXSTRLEN) [NUMBER\\_TO\\_CHARACTER\\_DP](#) (NUMBER, FORMAT, ERR, ERROR)

#### 7.154.1 Detailed Description

Converts a number to its equivalent character string representation.

Definition at line 80 of file strings.f90.

#### 7.154.2 Member Function Documentation

##### 7.154.2.1 CHARACTER(LEN=MAXSTRLEN) STRINGS::NUMBER\_TO\_CHARACTER::NUMBER\_TO\_CHARACTER\_DP (REAL(DP),intent(in) NUMBER, CHARACTER(LEN=\*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]

###### Parameters:

**NUMBER** The number to convert

**FORMAT** The format to use in the conversion

**ERR** The error code

**ERROR** The error string

Definition at line 947 of file strings.f90.

##### 7.154.2.2 CHARACTER(LEN=MAXSTRLEN) STRINGS::NUMBER\_TO\_CHARACTER::NUMBER\_TO\_CHARACTER\_INTG (INTEGER(INTG),intent(in) NUMBER, CHARACTER(LEN=\*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]

###### Parameters:

**NUMBER** The number to convert

**FORMAT** The format to use in the conversion

**ERR** The error code

**ERROR** The error string

Definition at line 780 of file strings.f90.

**7.154.2.3 CHARACTER(LEN=MAXSTRLEN) STRINGS::NUMBER\_TO\_-  
CHARACTER::NUMBER\_TO\_CHARACTER\_LINTG (INTEGER(LINTG),intent(in)  
NUMBER, CHARACTER(LEN=\*),intent(in) FORMAT, INTEGER(INTG),intent(out)  
ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

**Parameters:**

**NUMBER** The number to convert

**FORMAT** The format to use in the conversion

**ERR** The error code

**ERROR** The error string

Definition at line 817 of file strings.f90.

**7.154.2.4 CHARACTER(LEN=MAXSTRLEN) STRINGS::NUMBER\_TO\_-  
CHARACTER::NUMBER\_TO\_CHARACTER\_SP (REAL(SP),intent(in) NUMBER,  
CHARACTER(LEN=\*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR,  
TYPE(VARYING\_STRING),intent(out) ERROR) [private]**

**Parameters:**

**NUMBER** The number to convert

**FORMAT** The format to use in the conversion

**ERR** The error code

**ERROR** The error string

Definition at line 854 of file strings.f90.

## 7.155 STRINGS::NUMBER\_TO\_VSTRING Interface Reference

Converts a number to its equivalent varying string representation.

### Private Member Functions

- TYPE(VARYING\_STRING) NUMBER\_TO\_VSTRING\_INTG (NUMBER, FORMAT, ERR, ERROR)
- TYPE(VARYING\_STRING) NUMBER\_TO\_VSTRING\_LINTG (NUMBER, FORMAT, ERR, ERROR)
- TYPE(VARYING\_STRING) NUMBER\_TO\_VSTRING\_SP (NUMBER, FORMAT, ERR, ERROR)
- TYPE(VARYING\_STRING) NUMBER\_TO\_VSTRING\_DP (NUMBER, FORMAT, ERR, ERROR)

#### 7.155.1 Detailed Description

Converts a number to its equivalent varying string representation.

Definition at line 88 of file strings.f90.

#### 7.155.2 Member Function Documentation

**7.155.2.1** TYPE(VARYING\_STRING) STRINGS::NUMBER\_TO\_VSTRING::NUMBER\_TO\_VSTRING\_DP (REAL(DP),intent(in) *NUMBER*, CHARACTER(LEN=\*),intent(in) *FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]

##### Parameters:

*NUMBER* The number to convert

*FORMAT* The format to use in the conversion

*ERR* The error code

*ERROR* The error string

Definition at line 1222 of file strings.f90.

**7.155.2.2** TYPE(VARYING\_STRING) STRINGS::NUMBER\_TO\_VSTRING::NUMBER\_TO\_VSTRING\_INTG (INTEGER(INTG),intent(in) *NUMBER*, CHARACTER(LEN=\*),intent(in) *FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]

##### Parameters:

*NUMBER* The number to convert

*FORMAT* The format to use in the conversion

*ERR* The error code

*ERROR* The error string

Definition at line 1039 of file strings.f90.

**7.155.2.3** **TYPE(VARYING\_STRING) STRINGS::NUMBER\_TO\_VSTRING::NUMBER\_TO\_VSTRING\_LINTG (INTEGER(LINTG),intent(in) *NUMBER*, CHARACTER(LEN=\*),intent(in) *FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]**

**Parameters:**

***NUMBER*** The number to convert

***FORMAT*** The format to use in the conversion

***ERR*** The error code

***ERROR*** The error string

Definition at line 1082 of file strings.f90.

**7.155.2.4** **TYPE(VARYING\_STRING) STRINGS::NUMBER\_TO\_VSTRING::NUMBER\_TO\_VSTRING\_SP (REAL(SP),intent(in) *NUMBER*, CHARACTER(LEN=\*),intent(in) *FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]**

**Parameters:**

***NUMBER*** The number to convert

***FORMAT*** The format to use in the conversion

***ERR*** The error code

***ERROR*** The error string

Definition at line 1125 of file strings.f90.

## 7.156 **STRINGS::STRING\_TO\_DOUBLE** Interface Reference

Converts a string representation of a number to a double precision number.

### Private Member Functions

- REAL(DP) [STRING\\_TO\\_DOUBLE\\_C](#) (STRING, ERR, ERROR)
- REAL(DP) [STRING\\_TO\\_DOUBLE\\_VS](#) (STRING, ERR, ERROR)

#### 7.156.1 Detailed Description

Converts a string representation of a number to a double precision number.

Definition at line 96 of file strings.f90.

#### 7.156.2 Member Function Documentation

##### 7.156.2.1 **REAL(DP) STRINGS::STRING\_TO\_DOUBLE::STRING\_TO\_DOUBLE\_C** (CHARACTER(LEN=\*),intent(in) STRING, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**) [private]

###### Parameters:

**STRING** The string to convert

**ERR** The error code !<The error code

**ERROR** The error string !<The error string

Definition at line 1322 of file strings.f90.

##### 7.156.2.2 **REAL(DP) STRINGS::STRING\_TO\_DOUBLE::STRING\_TO\_DOUBLE\_VS** (TYPE(VARYING\_STRING),intent(in) STRING, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING\_STRING),intent(out) **ERROR**) [private]

###### Parameters:

**STRING** The string to convert

**ERR** The error code

**ERROR** The error string

Definition at line 1349 of file strings.f90.

## 7.157 STRINGS::STRING\_TO\_INTEGER Interface Reference

Converts a string representation of a number to an integer.

### Private Member Functions

- INTEGER(INTG) [STRING\\_TO\\_INTEGER\\_C](#) (STRING, ERR, ERROR)
- INTEGER(INTG) [STRING\\_TO\\_INTEGER\\_VS](#) (STRING, ERR, ERROR)

#### 7.157.1 Detailed Description

Converts a string representation of a number to an integer.

Definition at line 102 of file strings.f90.

#### 7.157.2 Member Function Documentation

##### 7.157.2.1 INTEGER(INTG) STRINGS::STRING\_TO\_INTEGER::STRING\_TO\_INTEGER\_C (CHARACTER(LEN=\*),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]

###### Parameters:

**STRING** The string to convert

**ERR** The error code

**ERROR** The error string

Definition at line 1380 of file strings.f90.

##### 7.157.2.2 INTEGER(INTG) STRINGS::STRING\_TO\_INTEGER::STRING\_TO\_INTEGER\_VS (TYPE(VARYING\_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR) [private]

###### Parameters:

**STRING** The string to convert

**ERR** The error code

**ERROR** The error string

Definition at line 1407 of file strings.f90.

## 7.158 `STRINGS::STRING_TO_LOGICAL` Interface Reference

Converts a string representation of a boolean value (TRUE or FALSE) to a logical.

### Private Member Functions

- LOGICAL `STRING_TO_LOGICAL_C` (`STRING`, `ERR`, `ERROR`)
- LOGICAL `STRING_TO_LOGICAL_VS` (`STRING`, `ERR`, `ERROR`)

#### 7.158.1 Detailed Description

Converts a string representation of a boolean value (TRUE or FALSE) to a logical.

Definition at line 114 of file strings.f90.

#### 7.158.2 Member Function Documentation

##### 7.158.2.1 LOGICAL `STRINGS::STRING_TO_LOGICAL::STRING_TO_LOGICAL_C` (`CHARACTER(LEN=*)`,`intent(in)` `STRING`, `INTEGER(INTG)`,`intent(out)` `ERR`, `TYPE(VARYING_STRING)`,`intent(out)` `ERROR`) [private]

###### Parameters:

**`STRING`** The string to convert

**`ERR`** The error code

**`ERROR`** The error string

Definition at line 1496 of file strings.f90.

##### 7.158.2.2 LOGICAL `STRINGS::STRING_TO_LOGICAL::STRING_TO_LOGICAL_VS` (`TYPE(VARYING_STRING)`,`intent(in)` `STRING`, `INTEGER(INTG)`,`intent(out)` `ERR`, `TYPE(VARYING_STRING)`,`intent(out)` `ERROR`) [private]

###### Parameters:

**`STRING`** The string to convert

**`ERR`** The error code

**`ERROR`** The error string

Definition at line 1523 of file strings.f90.

## 7.159 STRINGS::STRING\_TO\_LONG\_INTEGER Interface Reference

Converts a string representation of a number to a long integer.

### Private Member Functions

- INTEGER(LINTG) [STRING\\_TO\\_LONG\\_INTEGER\\_C](#) (STRING, ERR, ERROR)
- INTEGER(LINTG) [STRING\\_TO\\_LONG\\_INTEGER\\_VS](#) (STRING, ERR, ERROR)

### 7.159.1 Detailed Description

Converts a string representation of a number to a long integer.

Definition at line 108 of file strings.f90.

### 7.159.2 Member Function Documentation

**7.159.2.1 INTEGER(LINTG) STRINGS::STRING\_TO\_LONG\_INTEGER::STRING\_TO\_LONG\_INTEGER\_C (CHARACTER(LEN=\*),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR)  
[private]**

#### Parameters:

**STRING** The string to convert

**ERR** The error code

**ERROR** The error string

Definition at line 1438 of file strings.f90.

**7.159.2.2 INTEGER(LINTG) STRINGS::STRING\_TO\_LONG\_INTEGER::STRING\_TO\_LONG\_INTEGER\_VS (TYPE(VARYING\_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING\_STRING),intent(out) ERROR)  
[private]**

#### Parameters:

**STRING** The string to convert

**ERR** The error code

**ERROR** The error string

Definition at line 1465 of file strings.f90.

## 7.160 **STRINGS::STRING\_TO\_SINGLE** Interface Reference

Converts a string representation of a number to a single precision number.

### Private Member Functions

- REAL(SP) **STRING\_TO\_SINGLE\_C** (STRING, ERR, ERROR)
- REAL(SP) **STRING\_TO\_SINGLE\_VS** (STRING, ERR, ERROR)

#### 7.160.1 Detailed Description

Converts a string representation of a number to a single precision number.

Definition at line 120 of file strings.f90.

#### 7.160.2 Member Function Documentation

##### 7.160.2.1 **REAL(SP) STRINGS::STRING\_TO\_SINGLE::STRING\_TO\_SINGLE\_C** (CHARACTER(LEN=\*),intent(in) *STRING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]

###### Parameters:

**STRING** The string to convert

**ERR** The error code

**ERROR** The error string

Definition at line 1552 of file strings.f90.

##### 7.160.2.2 **REAL(SP) STRINGS::STRING\_TO\_SINGLE::STRING\_TO\_SINGLE\_VS** (TYPE(VARYING\_STRING),intent(in) *STRING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING\_STRING),intent(out) *ERROR*) [private]

###### Parameters:

**STRING** The string to convert

**ERR** The error code

**ERROR** The error string

Definition at line 1579 of file strings.f90.

## 7.161 STRINGS::VSTRING\_TO\_LOWERCASE Interface Reference

Returns a varying string which is the lowercase equivalent of the supplied string.

### Private Member Functions

- TYPE(VARYING\_STRING) VSTRING\_TO\_LOWERCASE\_C (STRING)
- TYPE(VARYING\_STRING) VSTRING\_TO\_LOWERCASE\_VS (STRING)

#### 7.161.1 Detailed Description

Returns a varying string which is the lowercase equivalent of the supplied string.

Definition at line 132 of file strings.f90.

#### 7.161.2 Member Function Documentation

**7.161.2.1** TYPE(VARYING\_STRING) STRINGS::VSTRING\_TO\_LOWERCASE::VSTRING\_TO\_LOWERCASE\_C (CHARACTER(LEN=\*)*,intent(in)* STRING)  
[private]

##### Parameters:

*STRING* The string to convert

Definition at line 1659 of file strings.f90.

**7.161.2.2** TYPE(VARYING\_STRING) STRINGS::VSTRING\_TO\_LOWERCASE::VSTRING\_TO\_LOWERCASE\_VS (TYPE(VARYING\_STRING)*,intent(in)* STRING)  
[private]

##### Parameters:

*STRING* The string to convert

Definition at line 1684 of file strings.f90.

## 7.162 **STRINGS::VSTRING\_TO\_UPPERCASE** Interface Reference

Returns a varying string which is the uppercase equivalent of the supplied string.

### Private Member Functions

- TYPE(VARYING\_STRING) VSTRING\_TO\_UPPERCASE\_C (STRING)
- TYPE(VARYING\_STRING) VSTRING\_TO\_UPPERCASE\_VS (STRING)

#### 7.162.1 Detailed Description

Returns a varying string which is the uppercase equivalent of the supplied string.

Definition at line 144 of file strings.f90.

#### 7.162.2 Member Function Documentation

##### 7.162.2.1 TYPE(VARYING\_STRING) STRINGS::VSTRING\_TO\_UPPERCASE::VSTRING\_TO\_UPPERCASE\_C (CHARACTER(LEN=\*),intent(in) STRING) [private]

###### Parameters:

*STRING* The string to convert

Definition at line 1759 of file strings.f90.

##### 7.162.2.2 TYPE(VARYING\_STRING) STRINGS::VSTRING\_TO\_UPPERCASE::VSTRING\_TO\_UPPERCASE\_VS (TYPE(VARYING\_STRING),intent(in) STRING) [private]

###### Parameters:

*STRING* The string to convert

Definition at line 1784 of file strings.f90.

## 7.163 TIMER::interface Interface Reference

### Private Member Functions

- subroutine [CPUTIMER](#) (RETURN\_TIME, TIME\_TYPE, ERR, CERROR)

#### 7.163.1 Detailed Description

Definition at line 69 of file timer\_f.f90.

#### 7.163.2 Member Function Documentation

**7.163.2.1 subroutine TIMER::interface::CPUTIMER (REAL(DP),intent(out) *RETURN\_TIME*,  
INTEGER(INTG),intent(in) *TIME\_TYPE*, INTEGER(INTG),intent(out) *ERR*,  
INTEGER(INTG),dimension(\*),intent(out) *CERROR*) [private]**

Definition at line 71 of file timer\_f.f90.

References TIMER::CPU\_TIMER().

Here is the call graph for this function:

## 7.164 TREES::TREE\_NODE\_TYPE Struct Reference

Collaboration diagram for TREES::TREE\_NODE\_TYPE:

### Private Attributes

- INTEGER(INTG) **KEY**  
*The key for the tree node.*
- INTEGER(INTG) **VALUE**  
*The value stored at the tree node.*
- INTEGER(INTG) **COLOUR**  
*The colour of the node for the red-black tree.*
- TYPE(TREE\_NODE\_TYPE), pointer **LEFT**  
*The pointer to the left daughter tree node if any.*
- TYPE(TREE\_NODE\_TYPE), pointer **RIGHT**  
*The pointer to the right daughter tree node if any.*
- TYPE(TREE\_NODE\_TYPE), pointer **PARENT**  
*The pointer to the parent tree node.*

### 7.164.1 Detailed Description

Definition at line 84 of file trees.f90.

### 7.164.2 Member Data Documentation

#### 7.164.2.1 INTEGER(INTG) TREES::TREE\_NODE\_TYPE::COLOUR [private]

The colour of the node for the red-black tree.

Definition at line 88 of file trees.f90.

#### 7.164.2.2 INTEGER(INTG) TREES::TREE\_NODE\_TYPE::KEY [private]

The key for the tree node.

Definition at line 86 of file trees.f90.

#### 7.164.2.3 TYPE(TREE\_NODE\_TYPE),pointer TREES::TREE\_NODE\_TYPE::LEFT [private]

The pointer to the left daughter tree node if any.

Definition at line 89 of file trees.f90.

**7.164.2.4 TYPE(TREE\_NODE\_TYPE),pointer TREES::TREE\_NODE\_TYPE::PARENT  
[private]**

The pointer to the parent tree node.

Definition at line 91 of file trees.f90.

**7.164.2.5 TYPE(TREE\_NODE\_TYPE),pointer TREES::TREE\_NODE\_TYPE::RIGHT  
[private]**

The pointer to the right daughter tree node if any.

Definition at line 90 of file trees.f90.

**7.164.2.6 INTEGER(INTG) TREES::TREE\_NODE\_TYPE::VALUE [private]**

The value stored at the tree node.

Definition at line 87 of file trees.f90.

## 7.165 TREES::TREE\_TYPE Struct Reference

Collaboration diagram for TREES::TREE\_TYPE:

### Private Attributes

- LOGICAL TREE\_FINISHED  
*Is .TRUE. if the tree has finished being created, .FALSE. if not.*
- INTEGER(INTG) INSERT\_TYPE  
*The insert type for duplicate keys for the tree.*
- INTEGER(INTG) NUMBER\_IN\_TREE  
*The number of items currently in the tree.*
- TYPE(TREE\_NODE\_TYPE), pointer ROOT  
*The pointer to the root of the tree.*
- TYPE(TREE\_NODE\_TYPE), pointer NIL  
*The pointer to the nil of the tree.*

### 7.165.1 Detailed Description

Definition at line 94 of file trees.f90.

### 7.165.2 Member Data Documentation

#### 7.165.2.1 INTEGER(INTG) TREES::TREE\_TYPE::INSERT\_TYPE [private]

The insert type for duplicate keys for the tree.

Definition at line 97 of file trees.f90.

#### 7.165.2.2 TYPE(TREE\_NODE\_TYPE),pointer TREES::TREE\_TYPE::NIL [private]

The pointer to the nil of the tree.

Definition at line 100 of file trees.f90.

#### 7.165.2.3 INTEGER(INTG) TREES::TREE\_TYPE::NUMBER\_IN\_TREE [private]

The number of items currently in the tree.

Definition at line 98 of file trees.f90.

#### 7.165.2.4 TYPE(TREE\_NODE\_TYPE),pointer TREES::TREE\_TYPE::ROOT [private]

The pointer to the root of the tree.

Definition at line 99 of file trees.f90.

**7.165.2.5 LOGICAL TREES::TREE\_TYPE::TREE\_FINISHED [private]**

Is .TRUE. if the tree has finished being created, .FALSE. if not.

Definition at line 96 of file trees.f90.

## 7.166 TYPES::BASIS\_FUNCTIONS\_TYPE Struct Reference

Contains information on the defined basis functions.

Collaboration diagram for TYPES::BASIS\_FUNCTIONS\_TYPE:

### Public Attributes

- INTEGER(INTG) [NUMBER\\_BASIS\\_FUNCTIONS](#)  
*The number of basis functions definegd.*
- TYPE([BASIS\\_PTR\\_TYPE](#)), pointer [BASES](#)  
*The array of pointers to the defined basis functions.*

### 7.166.1 Detailed Description

Contains information on the defined basis functions.

Definition at line 173 of file types.f90.

### 7.166.2 Member Data Documentation

#### 7.166.2.1 TYPE(BASIS\_PTR\_TYPE),pointer TYPES::BASIS\_FUNCTIONS\_TYPE::BASES

The array of pointers to the defined basis functions.

Definition at line 175 of file types.f90.

#### 7.166.2.2 INTEGER(INTG) TYPES::BASIS\_FUNCTIONS\_TYPE::NUMBER\_BASIS\_FUNCTIONS

The number of basis functions definegd.

Definition at line 174 of file types.f90.

## 7.167 TYPES::BASIS\_PTR\_TYPE Struct Reference

A buffer type to allow for an array of pointers to a BASIS\_TYPE.

Collaboration diagram for TYPES::BASIS\_PTR\_TYPE:

### Public Attributes

- TYPE(BASIS\_TYPE), pointer PTR

*The pointer to the basis.*

### 7.167.1 Detailed Description

A buffer type to allow for an array of pointers to a BASIS\_TYPE.

Definition at line 118 of file types.f90.

### 7.167.2 Member Data Documentation

#### 7.167.2.1 TYPE(BASIS\_TYPE),pointer TYPES::BASIS\_PTR\_TYPE::PTR

The pointer to the basis.

Definition at line 119 of file types.f90.

## 7.168 TYPES::BASIS\_TYPE Struct Reference

Contains all information about a basis .

Collaboration diagram for TYPES::BASIS\_TYPE:

### Public Attributes

- INTEGER(INTG) [USER\\_NUMBER](#)

*The user defined identifier for the basis. The user number must be unique.*

- INTEGER(INTG) [GLOBAL\\_NUMBER](#)

*The global number for the basis i.e., the position identifier for the list of bases defined.*

- INTEGER(INTG) [FAMILY\\_NUMBER](#)

*The family number for the basis. A basis has a number of sub-bases attached which make a basis family. The main parent basis is the basis defined by the user and it will have a family number of 0. The sub-bases of the parent basis will be the line and face bases that make up the basis. These will have different family numbers.*

- LOGICAL [BASIS\\_FINISHED](#)

*Is .TRUE. if the basis has finished being created, .FALSE. if not.*

- LOGICAL [HERMITE](#)

*Is .TRUE. if the basis is a hermite basis, .FALSE. if not.*

- INTEGER(INTG) [TYPE](#)

*The type of basis.*

- INTEGER(INTG) [NUMBER\\_OF\\_XI](#)

*The number of xi directions for the basis.*

- INTEGER(INTG) [NUMBER\\_OF\\_XI\\_COORDINATES](#)

*The number of xi coordinate directions for the basis. For Lagrange Hermite tensor product basis functions this is equal to the number of Xi directions. For simplex basis functions this is equal to the number of Xi directions + 1.*

- INTEGER(INTG), allocatable [INTERPOLATION\\_XI](#)

*INTERPOLATION\_XI(ni). The interpolation specification used in the ni'th Xi direction.*

- INTEGER(INTG), allocatable [INTERPOLATION\\_TYPE](#)

*INTERPOLATION\_TYPE(ni). The interpolation type in the ni'Xi coordinate direction. Old CMISS name IBT(1,ni,nb).*

- INTEGER(INTG), allocatable [INTERPOLATION\\_ORDER](#)

*INTERPOLATION\_ORDER(ni). The interpolation order in the ni'Xi coordinate direction. Old CMISS name IBT(2,ni,nb).*

- LOGICAL [DEGENERATE](#)

*Is .TRUE. if the basis is a degenerate basis (i.e., has collapsed nodes), .FALSE. if not.*

- INTEGER(INTG), allocatable **COLLAPSED\_XI**

*COLLAPSED\_XI(ni).* The collapsed state of the ni'th direction. *COLLAPSED\_XI* can be either *XI\_COLLAPSED*, *COLLAPSED\_AT\_XI0*, *COLLAPSED\_AT\_XII* or *NOT\_COLLAPSED* depending on whether or not the ni'th direction is collapsed, has a perpendicular Xi collapsed at the xi=0 end of the ni'th direction, has a perpendicular xi collapsed at the xi=1 of the ni'th direction or is not collapsed. NOTE: in old cmiss the value *IBT(1,ni) = 5* or *6* was set for the ni that was collapsed. The perpendicular line/face was then stored in *IBT(3,ni)*. For this the quadratic1 and quadratic2 type interpolation types are set on the perpendicular xi direction and the ni direction that is collapsed will have *COLLAPSED\_XI(ni)* set to *XI\_COLLAPSED*. BE CAREFUL WITH THIS WHEN TRANSLATING OLD **CMISS** CODE. Old **CMISS** name *IBT(1,ni) ????*

- INTEGER(INTG) **NUMBER\_OF\_COLLAPSED\_XI**

*The number of xi directions in the basis that are collapsed.*

- LOGICAL, allocatable **NODE\_AT\_COLLAPSE**

*NODE\_AT\_COLLAPSE(nn).* Is .TRUE. if the nn'th node of the basis is at a collapse, .FALSE. if not.

- TYPE(QUADRATURE\_TYPE) **QUADRATURE**

*The quadrature schemes for the basis.*

- INTEGER(INTG) **NUMBER\_OF\_PARTIAL\_DERIVATIVES**

*The number of paratial derivatives for the basis. Old **CMISS** name *NUT(nb)*.*

- INTEGER(INTG) **NUMBER\_OF\_NODES**

*The number of local nodes in the basis. Old **CMISS** name *NNT(nb)*.*

- INTEGER(INTG), allocatable **NUMBER\_OF\_NODES\_XI**

*NUMBER\_OF\_NODES\_XI(ni).* The number of local nodes in the ni'th direction in the basis. Old **CMISS** name *IBT(2,ni,nb)*.

- INTEGER(INTG) **NUMBER\_OF\_ELEMENT\_PARAMETERS**

*The number of element parameters in the basis. Old **CMISS** name *NST(nb)*.*

- INTEGER(INTG) **MAXIMUM\_NUMBER\_OF\_DERIVATIVES**

*The maximum number of derivatives at any node in the basis. Old **CMISS** name *NKT(0,nb)*.*

- INTEGER(INTG), allocatable **NUMBER\_OF\_DERIVATIVES**

*NUMBER\_OF\_DERIVATIVES(nn).* The number of derivatives at the nn'th node in the basis. Old **CMISS** name *NKT(nn,nb)*.

- INTEGER(INTG), allocatable **NODE\_POSITION\_INDEX**

*NODE\_POSITION\_INDEX(nn,nic).* The index of the node position for the nn'th local node in the nic'th coordinate. For Lagrange-Hermite tensor product basis functions: The number of coordinates equals the number of xi directions. Thus if *NODE\_POSITION\_INDEX(nn,:)=1,2,2* then local node nn is the first node in the ni(c)=1 direction, the second node in the ni(c)=2 direction and the second node in the ni(c)=3 direction; For simplex basis functions: The number of coordinates equals the number of xi directions plus one. The index specifies the inverse distance away from the corner/end of that area coordinate. Thus if an element has quadratic interpolation the index will range from 3 (closest to the corner/end of the element that the area coordinate has the value 1) to 1 (closest to the corners/end of the element that the area coordinate has the value 0). If M is the order of the element then in *NODE\_POSITION\_INDEX(nn,:)=1,1,M* then that node is apex for the third area coordinate. In general the index values will add up to M+number of xi directions+1 (i.e., subtract one from the indicies to get the standard simplex coordinates. Old **CMISS** name *INP(nn,ni,nb)*.

- INTEGER(INTG), allocatable [NODE\\_POSITION\\_INDEX\\_INV](#)

*NODE\_POSITION\_INDEX\_INV(nnc1,nnc2,nnc3,nnc4).* The inverse of the node position index for the basis. The NODE\_POSITION\_INDEX\_INV gives the local node number for the node that has node position indices of nnc1 in the 1st ni(c) direction, nnc2 in the 2nd ni(c) direction, nnc3 in the 3rd ni(c) direction and nnc4 in the 4th ni(c) direction. NOTE: that if the basis has less than 4 ni(c) direction the position index is 1. Old CMISS name NNB(inp1,inp2,inp3,nbf).

- INTEGER(INTG), allocatable [DERIVATIVE\\_ORDER\\_INDEX](#)

*DERIVATIVE\_ORDER\_INDEX(nk,nn,0:ni,nbf).* The index of the derivative order for the nk'th derivative of the nn'th node in the ni'th direction of the basis. The derivative index is NO\_PART\_DERIV for zeroth order, FIRST\_PART\_DERIV for the first order and SECOND\_PART\_DERIV for the second order derivative. Thus a DERIVATIVE\_ORDER\_INDEX(nk,nn,1..) of {NO\_PART\_DERIV,FIRST\_PART\_DERIV,NO\_PART\_DERIV} indicates that the nk'th derivative of the nn'th node of the basis is the first derivative with respect to the s2 direction. Old CMISS name IDO(nk,nn,1:ni,nbf).

- INTEGER(INTG), allocatable [DERIVATIVE\\_ORDER\\_INDEX\\_INV](#)

*DERIVATIVE\_ORDER\_INDEX\_INV(nu1,nu2,nu3,nn).* The inverse of the derivative order index for the nn'th local node of the basis. DERIVATIVE\_ORDER\_INDEX\_INV gives the derivative number for the nu1 partial derivative in the 1st xi direction, the nu2 partial derivative in the 2nd xi direction and the nu3 partial derivative in the 3rd xi direction. NOTE: local node nn does not carry any derivatives of the requested partial derivative type then DERIVATIVE\_ORDER\_INDEX\_INV will return 0. If the basis has less than 3 xi directions then the nu index is 1.

- INTEGER(INTG), allocatable [PARTIAL\\_DERIVATIVE\\_INDEX](#)

*PARTIAL\_DERIVATIVE\_INDEX(nk,nn).* Gives the partial derivative number (nu) of the nk'th derivative of the nn'th local node for the basis. Old CMISS name IDO(nk,nn,0,nbf).

- INTEGER(INTG), allocatable [ELEMENT\\_PARAMETER\\_INDEX](#)

*ELEMENT\_PARAMETER\_INDEX(nk,nn).* Gives the element parameter number (ns) of the nk'th derivative of the nn'th local node for the basis. Old CMISS name NSB(nk,nn,nbf).

- INTEGER(INTG) [NUMBER\\_OF\\_LOCAL\\_LINES](#)

*The number of local lines in the basis.*

- INTEGER(INTG), allocatable [LOCAL\\_LINE\\_XI\\_DIRECTION](#)

*LOCAL\_LINE\_XI\_DIRECTION(nae).* The Xi direction of the nae'th local line for the basis.

- INTEGER(INTG), allocatable [NUMBER\\_OF\\_NODES\\_IN\\_LOCAL\\_LINE](#)

*NUMBER\_OF\_NODES\_IN\_LOCAL\_LINE(nae).* The the number of nodes in the nae'th local line for the basis. Old CMISS name NNL(0,nae,nb).

- INTEGER(INTG), allocatable [NODE\\_NUMBERS\\_IN\\_LOCAL\\_LINE](#)

*NODE\_NUMBERS\_IN\_LOCAL\_LINE(nnl,nae).* The local node numbers (nn) for the nnl'th line node in the nae'th local line for the basis. Old CMISS name NNL(1..,nae,nb).

- INTEGER(INTG), allocatable [DERIVATIVE\\_NUMBERS\\_IN\\_LOCAL\\_LINE](#)

*DERIVATIVES\_NUMBERS\_IN\_LOCAL\_LINE(nnl,nae).* The derivative numbers (nk) for the nnl'th line node in the nae'th local line for the basis.

- [TYPE\(BASIS\\_PTR\\_TYPE\)](#), pointer [LINE\\_BASES](#)

*LINE\_BASES(nae).* The pointer to the basis for the nae'th line for the basis.

- [TYPE\(BASIS\\_PTR\\_TYPE\)](#), pointer [FACE\\_BASES](#)

*FACE\_BASES(naf).* The pointer to the basis for the naf'th face for the basis.

- [INTEGER\(INTG\) NUMBER\\_OF\\_SUB\\_BASES](#)

*The number of sub-bases (lines, faces) for the basis.*

- [TYPE\(BASIS\\_PTR\\_TYPE\)](#), pointer [SUB\\_BASES](#)

*SUB\_BASES(sbn).* The pointer to the sbn'th sub-basis for the basis.

- [TYPE\(BASIS\\_TYPE\)](#), pointer [PARENT\\_BASIS](#)

*The pointer to the parent basis for the basis. NOTE: that if the basis is not a sub-basis of another basis this pointer will be NULL.*

## 7.168.1 Detailed Description

Contains all information about a basis .

Definition at line 123 of file types.f90.

## 7.168.2 Member Data Documentation

### 7.168.2.1 LOGICAL TYPES::BASIS\_TYPE::BASIS\_FINISHED

Is .TRUE. if the basis has finished being created, .FALSE. if not.

Definition at line 128 of file types.f90.

### 7.168.2.2 INTEGER(INTG),allocatable TYPES::BASIS\_TYPE::COLLAPSED\_XI

COLLAPSED\_XI(ni). The collapsed state of the ni'th direction. COLLAPSED\_XI can be either XI\_COLLAPSED, COLLAPSED\_AT\_XI0, COLLAPSED\_AT\_XI1 or NOT\_COLLAPSED depending on whether or not the ni'th direction is collapsed, has a perpendicular Xi collapsed at the xi=0 end of the ni'th direction, has a perpendicular xi collapsed at the xi=1 of the ni'th direction or is not collapsed. NOTE: in old cmiss the value IBT(1,ni) = 5 or 6 was set for the ni that was collapsed. The perpendicular line/face was then stored in IBT(3,ni). For this the quadratic1 and quadratic2 type interpolation types are set on the perpendicular xi direction and the ni direction that is collapsed will have COLLAPSED\_XI(ni) set to XI\_COLLAPSED. BE CAREFUL WITH THIS WHEN TRANSLATING OLD CMISS CODE. Old CMISS name IBT(1,ni) ????

See also:

[BASIS\\_ROUTINES\\_XiCollapse](#)

Definition at line 138 of file types.f90.

### 7.168.2.3 LOGICAL TYPES::BASIS\_TYPE::DEGENERATE

Is .TRUE. if the basis is a degenerate basis (i.e., has collapsed nodes), .FALSE. if not.

Definition at line 137 of file types.f90.

#### 7.168.2.4 INTEGER(INTG),allocatable TYPES::BASIS\_TYPE::DERIVATIVE\_NUMBERS\_IN\_LOCAL\_LINE

DERIVATIVES\_NUMBERS\_IN\_LOCAL\_LINE(nnl,nae). The derivative numbers (nk) for the nnl'th line node in the nae'th local line for the basis.

Definition at line 161 of file types.f90.

#### 7.168.2.5 INTEGER(INTG),allocatable TYPES::BASIS\_TYPE::DERIVATIVE\_ORDER\_INDEX

DERIVATIVE\_ORDER\_INDEX(nk,nn,0:ni,nbf). The index of the derivative order for the nk'th derivative of the nn'th node in the ni'th direction of the basis. The derivative index is NO\_PART\_DERIV for zeroth order, FIRST\_PART\_DERIV for the first order and SECOND\_PART\_DERIV for the second order derivative. Thus a DERIVATIVE\_ORDER\_INDEX(nk,nn,1..) of {NO\_PART\_DERIV,FIRST\_PART\_DERIV,NO\_PART\_DERIV} indicates that the nk'th derivative of the nn'th node of the basis is the first derivative with respect to the s2 direction. Old CMISS name IDO(nk,nn,1:ni,nbf).

See also:

[TYPES::BASIS\\_TYPE::DERIVATIVE\\_ORDER\\_INDEX\\_INV](#),  
CONSTANTS\_PartialDerivativeConstants

Definition at line 152 of file types.f90.

#### 7.168.2.6 INTEGER(INTG),allocatable TYPES::BASIS\_TYPE::DERIVATIVE\_ORDER\_INDEX\_INV

DERIVATIVE\_ORDER\_INDEX\_INV(nu1,nu2,nu3,nn). The inverse of the derivative order index for the nn'th local node of the basis. DERIVATIVE\_ORDER\_INDEX\_INV gives the derivative number for the nu1 partial derivative in the 1st xi direction, the nu2 partial derivative in the 2nd xi direction and the nu3 partial derivative in the 3rd xi direction. NOTE: local node nn does not carry any derivatives of the requested partial derivative type then DERIVATIVE\_ORDER\_INDEX\_INV will return 0. If the basis has less than 3 xi directions then the nu index is 1.

See also:

[TYPES::BASIS\\_TYPE::DERIVATIVE\\_ORDER\\_INDEX](#)

Definition at line 153 of file types.f90.

#### 7.168.2.7 INTEGER(INTG),allocatable TYPES::BASIS\_TYPE::ELEMENT\_PARAMETER\_INDEX

ELEMENT\_PARAMETER\_INDEX(nk,nn). Gives the element parameter number (ns) of the nk'th derivative of the nn'th local node for the basis. Old CMISS name NSB(nk,nn,nbf).

Definition at line 155 of file types.f90.

#### 7.168.2.8 TYPE(BASIS\_PTR\_TYPE),pointer TYPES::BASIS\_TYPE::FACE\_BASES

FACE\_BASES(naf). The pointer to the basis for the naf'th face for the basis.

Definition at line 166 of file types.f90.

**7.168.2.9 INTEGER(INTG) TYPES::BASIS\_TYPE::FAMILY\_NUMBER**

The family number for the basis. A basis has a number of sub-bases attached which make a basis family. The main parent basis is the basis defined by the user and it will have a family number of 0. The sub-bases of the parent basis will be the line and face bases that make up the basis. These will have different family numbers.

Definition at line 127 of file types.f90.

**7.168.2.10 INTEGER(INTG) TYPES::BASIS\_TYPE::GLOBAL\_NUMBER**

The global number for the basis i.e., the position identifier for the list of bases defined.

Definition at line 126 of file types.f90.

**7.168.2.11 LOGICAL TYPES::BASIS\_TYPE::HERMITE**

Is .TRUE. if the basis is a hermite basis, .FALSE. if not.

Definition at line 129 of file types.f90.

**7.168.2.12 INTEGER(INTG),allocatable TYPES::BASIS\_TYPE::INTERPOLATION\_ORDER**

INTERPOLATION\_ORDER(ni). The interpolation order in the ni'th Xi coordinate direction. Old [CMISS](#) name IBT(2,ni,nb).

**See also:**

[BASIS\\_ROUTINES\\_InterpolationOrder](#)

Definition at line 135 of file types.f90.

**7.168.2.13 INTEGER(INTG),allocatable TYPES::BASIS\_TYPE::INTERPOLATION\_TYPE**

INTERPOLATION\_TYPE(ni). The interpolation type in the ni'th Xi coordinate direction. Old [CMISS](#) name IBT(1,ni,nb).

**See also:**

[BASIS\\_ROUTINES\\_InterpolationTypes](#)

Definition at line 134 of file types.f90.

**7.168.2.14 INTEGER(INTG),allocatable TYPES::BASIS\_TYPE::INTERPOLATION\_XI**

INTERPOLATION\_XI(ni). The interpolation specification used in the ni'th Xi direction.

**See also:**

[BASIS\\_ROUTINES\\_InterpolationSpecifications](#)

Definition at line 133 of file types.f90.

**7.168.2.15 TYPE(BASIS\_PTR\_TYPE),pointer TYPES::BASIS\_TYPE::LINE\_BASES**

LINE\_BASES(nae). The pointer to the basis for the nae'th line for the basis.

Definition at line 165 of file types.f90.

**7.168.2.16 INTEGER(INTG),allocatable TYPES::BASIS\_TYPE::LOCAL\_LINE\_XI\_-DIRECTION**

LOCAL\_LINE\_XI\_DIRECTION(nae). The Xi direction of the nae'th local line for the basis.

Definition at line 158 of file types.f90.

**7.168.2.17 INTEGER(INTG) TYPES::BASIS\_TYPE::MAXIMUM\_NUMBER\_OF\_-DERIVATIVES**

The maximum number of derivatives at any node in the basis. Old [CMISS](#) name NKT(0,nbf).

Definition at line 148 of file types.f90.

**7.168.2.18 LOGICAL,allocatable TYPES::BASIS\_TYPE::NODE\_AT\_COLLAPSE**

NODE\_AT\_COLLAPSE(nn). Is .TRUE. if the nn'th node of the basis is at a collapse, .FALSE. if not.

Definition at line 140 of file types.f90.

**7.168.2.19 INTEGER(INTG),allocatable TYPES::BASIS\_TYPE::NODE\_NUMBERS\_IN\_-LOCAL\_LINE**

NODE\_NUMBERS\_IN\_LOCAL\_LINE(nnl,nae). The local node numbers (nn) for the nnl'th line node in the nae'th local line for the basis. Old [CMISS](#) name NNL(1..,nae,nb).

Definition at line 160 of file types.f90.

**7.168.2.20 INTEGER(INTG),allocatable TYPES::BASIS\_TYPE::NODE\_POSITION\_INDEX**

NODE\_POSITION\_INDEX(nn,ni). The index of the node position for the nn'th local node in the ni'th coordinate. For Lagrange-Hermite tensor product basis functions: The number of coordinates equals the number of xi directions. Thus if NODE\_POSITION\_INDEX(nn,:)=1,2,2 then local node nn is the first node in the ni(c)=1 direction, the second node in the ni(c)=2 direction and the second node in the ni(c)=3 direction; For simplex basis functions: The number of coordinates equals the number of xi directions plus one. The index specifies the inverse distance away from the corner/end of that area coordinate. Thus if an element has quadratic interpolation the index will range from 3 (closest to the corner/end of the element that the area coordinate has the value 1) to 1 (closest to the corners/end of the element that the area coordinate has the value 0). If M is the order of the element then in NODE\_POSITION\_INDEX(nn,:)=1,1,M then that node is apex for the third area coordinate. In general the index values will add up to M+number of xi directions+1 (i.e., subtract one from the indices to get the standard simplex coordinates. Old [CMISS](#) name INP(nn,ni,nb).

**See also:**

[TYPES::BASIS\\_TYPE::NODE\\_POSITION\\_INDEX\\_INV](#).

Definition at line 150 of file types.f90.

**7.168.2.21 INTEGER(INTG),allocatable TYPES::BASIS\_TYPE::NODE\_POSITION\_INDEX\_INV**

NODE\_POSITION\_INDEX\_INV(nnc1,nnc2,nnc3,nnc4). The inverse of the node position index for the basis. The NODE\_POSITION\_INDEX\_INV gives the local node number for the node that has node position indices of nnc1 in the 1st ni(c) direction, nnc2 in the 2nd ni(c) direction, nnc3 in the 3rd ni(c) direction and nnc4 in the 4th ni(c) direction. NOTE: that if the basis has less than 4 ni(c) direction the position index is 1. Old [CMISS](#) name NNB(inp1,inp2,inp3,nbf).

See also:

[TYPES::BASIS\\_TYPE::NODE\\_POSITION\\_INDEX](#).

Definition at line 151 of file types.f90.

**7.168.2.22 INTEGER(INTG) TYPES::BASIS\_TYPE::NUMBER\_OF\_COLLAPSED\_XI**

The number of xi directions in the basis that are collapsed.

Definition at line 139 of file types.f90.

**7.168.2.23 INTEGER(INTG),allocatable TYPES::BASIS\_TYPE::NUMBER\_OF\_DERIVATIVES**

NUMBER\_OF\_DERIVATIVES(nn). The number of derivatives at the nn'th node in the basis. Old [CMISS](#) name NKT(nn,nbf).

Definition at line 149 of file types.f90.

**7.168.2.24 INTEGER(INTG) TYPES::BASIS\_TYPE::NUMBER\_OF\_ELEMENT\_PARAMETERS**

The number of element parameters in the basis. Old [CMISS](#) name NST(nbf).

Definition at line 147 of file types.f90.

**7.168.2.25 INTEGER(INTG) TYPES::BASIS\_TYPE::NUMBER\_OF\_LOCAL\_LINES**

The number of local lines in the basis.

Definition at line 157 of file types.f90.

**7.168.2.26 INTEGER(INTG) TYPES::BASIS\_TYPE::NUMBER\_OF\_NODES**

The number of local nodes in the basis. Old [CMISS](#) name NNT(nbf).

Definition at line 144 of file types.f90.

**7.168.2.27 INTEGER(INTG),allocatable TYPES::BASIS\_TYPE::NUMBER\_OF\_NODES\_IN\_LOCAL\_LINE**

NUMBER\_OF\_NODES\_IN\_LOCAL\_LINE(nae). The the number of nodes in the nae'th local line for the basis. Old [CMISS](#) name NNL(0,nae,nb).

Definition at line 159 of file types.f90.

### 7.168.2.28 INTEGER(INTG),allocatable TYPES::BASIS\_TYPE::NUMBER\_OF\_NODES\_XI

NUMBER\_OF\_NODES\_XI(ni). The number of local nodes in the ni'th direction in the basis. Old [CMISS](#) name IBT(2,ni,nb).

#### [Todo](#)

CHANGE TO NUMBER\_OF\_NODES\_XIC i.e., xi coordinate index not xi???

Definition at line 146 of file types.f90.

### 7.168.2.29 INTEGER(INTG) TYPES::BASIS\_TYPE::NUMBER\_OF\_PARTIAL\_DERIVATIVES

The number of paratial derivatives for the basis. Old [CMISS](#) name NUT(nbf).

Definition at line 143 of file types.f90.

### 7.168.2.30 INTEGER(INTG) TYPES::BASIS\_TYPE::NUMBER\_OF\_SUB\_BASES

The number of sub-bases (lines, faces) for the basis.

Definition at line 167 of file types.f90.

### 7.168.2.31 INTEGER(INTG) TYPES::BASIS\_TYPE::NUMBER\_OF\_XI

The number of xi directions for the basis.

Definition at line 131 of file types.f90.

### 7.168.2.32 INTEGER(INTG) TYPES::BASIS\_TYPE::NUMBER\_OF\_XI\_COORDINATES

The number of xi coordinate directions for the basis. For Lagrange Hermite tensor product basis functions this is equal to the number of Xi directions. For simplex basis functions this is equal to the number of Xi directions + 1.

Definition at line 132 of file types.f90.

### 7.168.2.33 TYPE(BASIS\_TYPE),pointer TYPES::BASIS\_TYPE::PARENT\_BASIS

The pointer to the parent basis for the basis. NOTE: that if the basis is not a sub-basis of another basis this pointer will be NULL.

Definition at line 169 of file types.f90.

### 7.168.2.34 INTEGER(INTG),allocatable TYPES::BASIS\_TYPE::PARTIAL\_DERIVATIVE\_INDEX

PARTIAL\_DERIVATIVE\_INDEX(nk,nn). Gives the partial derivative number (nu) of the nk'th derivative of the nn'th local node for the basis. Old [CMISS](#) name IDO(nk,nn,0,nbf).

Definition at line 154 of file types.f90.

**7.168.2.35 TYPE(QUADRATURE\_TYPE) TYPES::BASIS\_TYPE::QUADRATURE**

The quadrature schemes for the basis.

Definition at line 142 of file types.f90.

**7.168.2.36 TYPE(BASIS\_PTR\_TYPE),pointer TYPES::BASIS\_TYPE::SUB\_BASES**

SUB\_BASES(sbn). The pointer to the sbn'th sub-basis for the basis.

Definition at line 168 of file types.f90.

**7.168.2.37 INTEGER(INTG) TYPES::BASIS\_TYPE::TYPE**

The type of basis.

**See also:**

BASIS\_ROUTINES\_BasisTypes

Definition at line 130 of file types.f90.

**7.168.2.38 INTEGER(INTG) TYPES::BASIS\_TYPE::USER\_NUMBER**

The user defined identifier for the basis. The user number must be unique.

Definition at line 125 of file types.f90.

## 7.169 TYPES::COORDINATE\_SYSTEM\_TYPE Struct Reference

Contains information on a coordinate system.

### Public Attributes

- INTEGER(INTG) [USER\\_NUMBER](#)

*The user defined identifier for the coordinate. The user number must be unique.*

- LOGICAL [COORDINATE\\_SYSTEM\\_FINISHED](#)

*Is .TRUE. if the coordinate system has finished being created, .FALSE. if not.*

- INTEGER(INTG) [TYPE](#)

*The type of coordinate system. Old [CMISS](#) name ITYP10(nr).*

- INTEGER(INTG) [RADIAL\\_INTERPOLATION\\_TYPE](#)

*The type of radial interpolation type for non-rectangular cartesian systems. Old [CMISS](#) name JTYP10(nr).*

- INTEGER(INTG) [NUMBER\\_OF\\_DIMENSIONS](#)

*The number of dimensions for the coordinate system. Old [CMISS](#) name NJT.*

- REAL(DP) [FOCUS](#)

*The focus of the coordinate system for a prolate-spheriodal coordinate system.*

- REAL(DP) [ORIGIN](#)

*ORIGIN(nj). The nj'th component of the origin of the coordinate system wrt the global coordinate system.  
NOTE: maybe this should be wrt to the parent regions coordinate system - this would then go into the REGION type.*

- REAL(DP) [ORIENTATION](#)

*ORIENTATION(nj,mj). he orientation matrix for the orientation of the coordinate system wrt to the global coordinate system. NOTE: maybe this should be wrt to the parent regions coordinate system - this would then go into the REGION type.*

### 7.169.1 Detailed Description

Contains information on a coordinate system.

#### Todo

Have a list of coordinate systems and have a pointer in the coordinate\_system\_type back to the regions that use them.

Definition at line 185 of file types.f90.

## 7.169.2 Member Data Documentation

### 7.169.2.1 LOGICAL TYPES::COORDINATE\_SYSTEM\_TYPE::COORDINATE\_SYSTEM\_FINISHED

Is .TRUE. if the coordinate system has finished being created, .FALSE. if not.

Definition at line 187 of file types.f90.

### 7.169.2.2 REAL(DP) TYPES::COORDINATE\_SYSTEM\_TYPE::FOCUS

The focus of the coordinate system for a prolate-spheriodal coordinate system.

Definition at line 191 of file types.f90.

### 7.169.2.3 INTEGER(INTG) TYPES::COORDINATE\_SYSTEM\_TYPE::NUMBER\_OF\_DIMENSIONS

The number of dimensions for the coordinate system. Old [CMISS](#) name NJT.

Definition at line 190 of file types.f90.

### 7.169.2.4 REAL(DP) TYPES::COORDINATE\_SYSTEM\_TYPE::ORIENTATION

ORIENTATION(nj,mj). he orientation matrix for the orientation of the coordinate system wrt to the global coordinate system. NOTE: maybe this should be wrt to the parent regions coordinate system - this would then go into the REGION type.

Definition at line 193 of file types.f90.

### 7.169.2.5 REAL(DP) TYPES::COORDINATE\_SYSTEM\_TYPE::ORIGIN

ORIGIN(nj). The nj'th component of the origin of the coordinate system wrt the global coordinate system. NOTE: maybe this should be wrt to the parent regions coordinate system - this would then go into the REGION type.

Definition at line 192 of file types.f90.

### 7.169.2.6 INTEGER(INTG) TYPES::COORDINATE\_SYSTEM\_TYPE::RADIAL\_INTERPOLATION\_TYPE

The type of radial interpolation type for non-rectangular cartesian systems. Old [CMISS](#) name JTYP10(nr).

See also:

[COORDINATE\\_ROUTINES::RadialInterpolations](#)

Definition at line 189 of file types.f90.

### 7.169.2.7 INTEGER(INTG) TYPES::COORDINATE\_SYSTEM\_TYPE::TYPE

The type of coordinate system. Old [CMISS](#) name ITYP10(nr).

**See also:**

[COORDINATE\\_ROUTINES::CoordinateSystemTypes](#)

Definition at line 188 of file types.f90.

**7.169.2.8 INTEGER(INTG) TYPES::COORDINATE\_SYSTEM\_TYPE::USER\_NUMBER**

The user defined identifier for the coordinate. The user number must be unique.

Definition at line 186 of file types.f90.

## 7.170 TYPES::DECOMPOSITION\_ELEMENT\_TYPE Struct Reference

Contains the information for an element in a decomposition.

### Public Attributes

- INTEGER(INTG) [LOCAL\\_NUMBER](#)

*The local element number in the decomposition.*

- INTEGER(INTG) [GLOBAL\\_NUMBER](#)

*The corresponding global element number in the mesh of the local element number in the decomposition.*

- INTEGER(INTG) [USER\\_NUMBER](#)

*The corresponding user number for the element.*

- INTEGER(INTG), allocatable [NUMBER\\_OF\\_ADJACENT\\_ELEMENTS](#)

*NUMBER\_OF\_ADJACENT\_ELEMENTS(-ni:ni). The number of elements adjacent to this element in the ni'th xi direction. Note that -ni gives the adjacent element before the element in the ni'th direction and +ni gives the adjacent element after the element in the ni'th direction. The ni=0 index should be 1 for the current element. Old CMISS name NXI(-ni:ni,0:nei,ne).*

- INTEGER(INTG), allocatable [ADJACENT\\_ELEMENTS](#)

*ADJACENT\_ELEMENTS(nei,-ni:ni). The local element numbers of the elements adjacent to this element in the ni'th xi direction. Note that -ni gives the adjacent elements before the element in the ni'th direction and +ni gives the adjacent elements after the element in the ni'th direction. The ni=0 index should give the current element number. Old CMISS name NXI(-ni:ni,0:nei,ne).*

- INTEGER(INTG), allocatable [ELEMENT\\_LINES](#)

*ELEMENT\_LINES(nae). The local decomposition line number corresponding to the nae'th local line of the element. Old CMISS name NLL(nae,ne).*

### 7.170.1 Detailed Description

Contains the information for an element in a decomposition.

Definition at line 697 of file types.f90.

### 7.170.2 Member Data Documentation

#### 7.170.2.1 INTEGER(INTG),allocatable TYPES::DECOMPOSITION\_ELEMENT\_TYPE::ADJACENT\_ELEMENTS

*ADJACENT\_ELEMENTS(nei,-ni:ni). The local element numbers of the elements adjacent to this element in the ni'th xi direction. Note that -ni gives the adjacent elements before the element in the ni'th direction and +ni gives the adjacent elements after the element in the ni'th direction. The ni=0 index should give the current element number. Old CMISS name NXI(-ni:ni,0:nei,ne).*

Definition at line 702 of file types.f90.

**7.170.2.2 INTEGER(INTG),allocatable TYPES::DECOMPOSITION\_ELEMENT\_-  
TYPE::ELEMENT\_LINES**

ELEMENT\_LINES(nae). The local decomposition line number corresponding to the nae'th local line of the element. Old [CMISS](#) name NLL(nae,ne).

Definition at line 703 of file types.f90.

**7.170.2.3 INTEGER(INTG) TYPES::DECOMPOSITION\_ELEMENT\_TYPE::GLOBAL\_-  
NUMBER**

The corresponding global element number in the mesh of the local element number in the decomposition.

Definition at line 699 of file types.f90.

**7.170.2.4 INTEGER(INTG) TYPES::DECOMPOSITION\_ELEMENT\_TYPE::LOCAL\_-  
NUMBER**

The local element number in the decomposition.

Definition at line 698 of file types.f90.

**7.170.2.5 INTEGER(INTG),allocatable TYPES::DECOMPOSITION\_ELEMENT\_-  
TYPE::NUMBER\_OF\_ADJACENT\_ELEMENTS**

NUMBER\_OF\_ADJACENT\_ELEMENTS(-ni:ni). The number of elements adjacent to this element in the ni'th xi direction. Note that -ni gives the adjacent element before the element in the ni'th direction and +ni gives the adjacent element after the element in the ni'th direction. The ni=0 index should be 1 for the current element. Old [CMISS](#) name NXI(-ni:ni,0:nei,ne).

Definition at line 701 of file types.f90.

**7.170.2.6 INTEGER(INTG) TYPES::DECOMPOSITION\_ELEMENT\_TYPE::USER\_NUMBER**

The corresponding user number for the element.

Definition at line 700 of file types.f90.

## 7.171 TYPES::DECOMPOSITION\_ELEMENTS\_TYPE Struct Reference

Contains the topology information for the elements of a decomposition.

Collaboration diagram for TYPES::DECOMPOSITION\_ELEMENTS\_TYPE:

### Public Attributes

- TYPE(DECOMPOSITION\_TYPE), pointer DECOMPOSITION

*The pointer to the decomposition for this elements topology information.*

- INTEGER(INTG) TOTAL\_NUMBER\_OF\_ELEMENTS

*The total number of elements in this decomposition topology.*

- TYPE(DECOMPOSITION\_ELEMENT\_TYPE), pointer ELEMENTS

*ELEMENTS(ne). The pointer to the array of topology information for the elements of this decomposition. ELEMENTS(ne) contains the topological information for the ne'th local element of the decomposition.*

### 7.171.1 Detailed Description

Contains the topology information for the elements of a decomposition.

Definition at line 707 of file types.f90.

### 7.171.2 Member Data Documentation

#### 7.171.2.1 TYPE(DECOMPOSITION\_TYPE),pointer TYPES::DECOMPOSITION\_ELEMENTS\_TYPE::DECOMPOSITION

The pointer to the decomposition for this elements topology information.

Definition at line 708 of file types.f90.

#### 7.171.2.2 TYPE(DECOMPOSITION\_ELEMENT\_TYPE),pointer TYPES::DECOMPOSITION\_ELEMENTS\_TYPE::ELEMENTS

ELEMENTS(ne). The pointer to the array of topology information for the elements of this decomposition. ELEMENTS(ne) contains the topological information for the ne'th local element of the decomposition.

#### **Todo**

Change this to allocatable???

Definition at line 710 of file types.f90.

**7.171.2.3 INTEGER(INTG) TYPES::DECOMPOSITION\_ELEMENTS\_TYPE::TOTAL\_-  
NUMBER\_OF\_ELEMENTS**

The total number of elements in this decomposition topology.

Definition at line 709 of file types.f90.

## 7.172 TYPES::DECOMPOSITION\_LINE\_TYPE Struct Reference

Contains the information for a line in a decomposition.

### Public Attributes

- INTEGER(INTG) **NUMBER**  
*The line number in the decomposition.*
- INTEGER(INTG) **XI\_DIRECTION**  
*The Xi direction of the line. Old CMISS name NPL(1,0,nl).*
- INTEGER(INTG) **NUMBER\_OF\_SURROUNDING\_ELEMENTS**  
*The number of elements that surround (use) this line.*
- INTEGER(INTG), allocatable **SURROUNDING\_ELEMENTS**  
*SURROUNDING\_ELEMENTS(nel). The local element number of the nel'th element that surrounds (uses) this line.*
- INTEGER(INTG), allocatable **ELEMENT\_LINES**  
*ELEMENT\_LINES(nel). The local arc number of the nel'th element that surrounds (uses) this line.*
- INTEGER(INTG) **ADJACENT\_LINES**  
*ADJACENT\_LINES(0:1). The line number of adjacent lines. ADJACENT\_LINES(0) is the line number adjacent in the -xi direction. ADJACENT\_LINES(1) is the line number adjacent in the +xi direction. Old CMISS name NPL(2..3,0,nl).*

### 7.172.1 Detailed Description

Contains the information for a line in a decomposition.

Definition at line 680 of file types.f90.

### 7.172.2 Member Data Documentation

#### 7.172.2.1 INTEGER(INTG) TYPES::DECOMPOSITION\_LINE\_TYPE::ADJACENT\_LINES

ADJACENT\_LINES(0:1). The line number of adjacent lines. ADJACENT\_LINES(0) is the line number adjacent in the -xi direction. ADJACENT\_LINES(1) is the line number adjacent in the +xi direction. Old CMISS name NPL(2..3,0,nl).

Definition at line 686 of file types.f90.

#### 7.172.2.2 INTEGER(INTG),allocatable TYPES::DECOMPOSITION\_LINE\_TYPE::ELEMENT\_LINES

ELEMENT\_LINES(nel). The local arc number of the nel'th element that surrounds (uses) this line.

Definition at line 685 of file types.f90.

**7.172.2.3 INTEGER(INTG) TYPES::DECOMPOSITION\_LINE\_TYPE::NUMBER**

The line number in the decomposition.

Definition at line 681 of file types.f90.

**7.172.2.4 INTEGER(INTG) TYPES::DECOMPOSITION\_LINE\_TYPE::NUMBER\_OF\_-  
SURROUNDING\_ELEMENTS**

The number of elements that surround (use) this line.

Definition at line 683 of file types.f90.

**7.172.2.5 INTEGER(INTG),allocatable TYPES::DECOMPOSITION\_LINE\_-  
TYPE::SURROUNDING\_ELEMENTS**

SURROUNDING\_ELEMENTS(nel). The local element number of the nel'th element that surrounds (uses) this line.

Definition at line 684 of file types.f90.

**7.172.2.6 INTEGER(INTG) TYPES::DECOMPOSITION\_LINE\_TYPE::XI\_DIRECTION**

The Xi direction of the line. Old [CMISS](#) name NPL(1,0,nl).

Definition at line 682 of file types.f90.

## 7.173 TYPES::DECOMPOSITION\_LINES\_TYPE Struct Reference

Contains the topology information for the lines of a decomposition.

Collaboration diagram for TYPES::DECOMPOSITION\_LINES\_TYPE:

### Public Attributes

- TYPE(DECOMPOSITION\_TYPE), pointer DECOMPOSITION  
*The pointer to the decomposition for this lines topology information.*
- INTEGER(INTG) NUMBER\_OF\_LINES  
*The number of lines in this decomposition topology.*
- TYPE(DECOMPOSITION\_LINE\_TYPE), allocatable LINES  
*LINES(nl). The pointer to the array of topology information for the lines of this decomposition. LINES(nl) contains the topological information for the nl'th local line of the decomposition.*

### 7.173.1 Detailed Description

Contains the topology information for the lines of a decomposition.

Definition at line 690 of file types.f90.

### 7.173.2 Member Data Documentation

#### 7.173.2.1 TYPE(DECOMPOSITION\_TYPE),pointer TYPES::DECOMPOSITION\_LINES\_TYPE::DECOMPOSITION

The pointer to the decomposition for this lines topology information.

Definition at line 691 of file types.f90.

#### 7.173.2.2 TYPE(DECOMPOSITION\_LINE\_TYPE),allocatable TYPES::DECOMPOSITION\_LINES\_TYPE::LINES

LINES(nl). The pointer to the array of topology information for the lines of this decomposition. LINES(nl) contains the topological information for the nl'th local line of the decomposition.

Definition at line 693 of file types.f90.

#### 7.173.2.3 INTEGER(INTG) TYPES::DECOMPOSITION\_LINES\_TYPE::NUMBER\_OF\_LINES

The number of lines in this decomposition topology.

Definition at line 692 of file types.f90.

## 7.174 TYPES::DECOMPOSITION\_PTR\_TYPE Struct Reference

A buffer type to allow for an array of pointers to a [DECOMPOSITION\\_TYPE](#).

Collaboration diagram for TYPES::DECOMPOSITION\_PTR\_TYPE:

### Public Attributes

- TYPE([DECOMPOSITION\\_TYPE](#)), pointer PTR

*The pointer to the domain decomposition.*

#### 7.174.1 Detailed Description

A buffer type to allow for an array of pointers to a [DECOMPOSITION\\_TYPE](#).

Definition at line 737 of file types.f90.

#### 7.174.2 Member Data Documentation

##### 7.174.2.1 TYPE(DECOMPOSITION\_TYPE),pointer TYPES::DECOMPOSITION\_PTR\_- TYPE::PTR

The pointer to the domain decomposition.

Definition at line 738 of file types.f90.

## 7.175 TYPES::DECOMPOSITION\_TOPOLOGY\_TYPE Struct Reference

Contains the topology information for a decomposition.

Collaboration diagram for TYPES::DECOMPOSITION\_TOPOLOGY\_TYPE:

### Public Attributes

- TYPE(DECOMPOSITION\_TYPE), pointer DECOMPOSITION  
*The pointer to the decomposition for this topology information.*
- TYPE(DECOMPOSITION\_ELEMENTS\_TYPE), pointer ELEMENTS  
*The pointer to the topology information for the elements of this decomposition.*
- TYPE(DECOMPOSITION\_LINES\_TYPE), pointer LINES  
*The pointer to the topology information for the lines of this decomposition.*

### 7.175.1 Detailed Description

Contains the topology information for a decomposition.

Definition at line 714 of file types.f90.

### 7.175.2 Member Data Documentation

#### 7.175.2.1 TYPE(DECOMPOSITION\_TYPE),pointer TYPES::DECOMPOSITION\_TOPOLOGY\_TYPE::DECOMPOSITION

The pointer to the decomposition for this topology information.

Definition at line 715 of file types.f90.

#### 7.175.2.2 TYPE(DECOMPOSITION\_ELEMENTS\_TYPE),pointer TYPES::DECOMPOSITION\_TOPOLOGY\_TYPE::ELEMENTS

The pointer to the topology information for the elements of this decomposition.

Definition at line 716 of file types.f90.

#### 7.175.2.3 TYPE(DECOMPOSITION\_LINES\_TYPE),pointer TYPES::DECOMPOSITION\_TOPOLOGY\_TYPE::LINES

The pointer to the topology information for the lines of this decomposition.

Definition at line 717 of file types.f90.

## 7.176 TYPES::DECOMPOSITION\_TYPE Struct Reference

Contains information on the domain decomposition.

Collaboration diagram for TYPES::DECOMPOSITION\_TYPE:

### Public Attributes

- INTEGER(INTG) [USER\\_NUMBER](#)

*The user defined identifier for the domain decomposition. The user number must be unique.*

- INTEGER(INTG) [GLOBAL\\_NUMBER](#)

*The global number of the domain decomposition in the list of domain decompositions for a particular mesh.*

- LOGICAL [DECOMPOSITION\\_FINISHED](#)

*Is .TRUE. if the decomposition has finished being created, .FALSE. if not.*

- TYPE([DECOMPOSITIONS\\_TYPE](#)), pointer [DECOMPOSITIONS](#)

*A pointer to the decompositions for this decomposition.*

- TYPE([MESH\\_TYPE](#)), pointer [MESH](#)

*A pointer to the mesh for this decomposition.*

- INTEGER(INTG) [MESH\\_COMPONENT\\_NUMBER](#)

*The component number (index) of the mesh component that this decomposition belongs to (i.e., was generated from).*

- INTEGER(INTG) [DECOMPOSITION\\_TYPE](#)

*The type of the domain decomposition.*

- INTEGER(INTG) [NUMBER\\_OF\\_DOMAINS](#)

*The number of domains that this decomposition contains.*

- INTEGER(INTG) [NUMBER\\_OF\\_EDGES\\_CUT](#)

*For automatically calculated decompositions, the number of edges of the mesh dual graph that were cut for the composition. It provides an indication of the optimality of the automatic decomposition.*

- INTEGER(INTG), allocatable [ELEMENT\\_DOMAIN](#)

*ELEMENT\_DOMAIN(ne). The domain number that the ne'th global element is in for the decomposition.  
Note: the domain numbers start at 0 and go up to the NUMBER\_OF\_DOMAINS-1.*

- TYPE([DECOMPOSITION\\_TOPOLOGY\\_TYPE](#)), pointer [TOPOLOGY](#)

*A pointer to the topology for this decomposition.*

- TYPE([DOMAIN\\_PTR\\_TYPE](#)), pointer [DOMAIN](#)

*DOMAIN(mesh\_component\_idx). A pointer to the domain for mesh component for the domain associated with the computational node.*

### 7.176.1 Detailed Description

Contains information on the domain decomposition.

Definition at line 721 of file types.f90.

### 7.176.2 Member Data Documentation

#### 7.176.2.1 LOGICAL TYPES::DECOMPOSITION\_TYPE::DECOMPOSITION\_FINISHED

Is .TRUE. if the decomposition has finished being created, .FALSE. if not.

Definition at line 724 of file types.f90.

#### 7.176.2.2 INTEGER(INTG) TYPES::DECOMPOSITION\_TYPE::DECOMPOSITION\_TYPE

The type of the domain decomposition.

See also:

[MESH\\_ROUTINES::DecompositionTypes](#).

Definition at line 728 of file types.f90.

#### 7.176.2.3 TYPE(DECOMPOSITIONS\_TYPE),pointer TYPES::DECOMPOSITION\_TYPE::DECOMPOSITIONS

A pointer to the decompositions for this decomposition.

Definition at line 725 of file types.f90.

#### 7.176.2.4 TYPE(DOMAIN\_PTR\_TYPE),pointer TYPES::DECOMPOSITION\_TYPE::DOMAIN

DOMAIN(mesh\_component\_idx). A pointer to the domain for mesh component for the domain associated with the computational node.

##### **Todo**

Change this to allocatable???

Definition at line 733 of file types.f90.

#### 7.176.2.5 INTEGER(INTG),allocatable TYPES::DECOMPOSITION\_TYPE::ELEMENT\_DOMAIN

ELEMENT\_DOMAIN(ne). The domain number that the ne'th global element is in for the decomposition.  
Note: the domain numbers start at 0 and go up to the NUMBER\_OF\_DOMAINS-1.

Definition at line 731 of file types.f90.

**7.176.2.6 INTEGER(INTG) TYPES::DECOMPOSITION\_TYPE::GLOBAL\_NUMBER**

The global number of the domain decomposition in the list of domain decompositions for a particular mesh.  
Definition at line 723 of file types.f90.

**7.176.2.7 TYPE(MESH\_TYPE),pointer TYPES::DECOMPOSITION\_TYPE::MESH**

A pointer to the mesh for this decomposition.  
Definition at line 726 of file types.f90.

**7.176.2.8 INTEGER(INTG) TYPES::DECOMPOSITION\_TYPE::MESH\_COMPONENT\_-  
NUMBER**

The component number (index) of the mesh component that this decomposition belongs to (i.e., was generated from).  
Definition at line 727 of file types.f90.

**7.176.2.9 INTEGER(INTG) TYPES::DECOMPOSITION\_TYPE::NUMBER\_OF\_DOMAINS**

The number of domains that this decomposition contains.  
Definition at line 729 of file types.f90.

**7.176.2.10 INTEGER(INTG) TYPES::DECOMPOSITION\_TYPE::NUMBER\_OF\_EDGES\_-  
CUT**

For automatically calcualted decompositions, the number of edges of the mesh dual graph that were cut for the composition. It provides an indication of the optimally of the automatic decomposition.  
Definition at line 730 of file types.f90.

**7.176.2.11 TYPE(DECOMPOSITION\_TOPOLOGY\_TYPE),pointer  
TYPES::DECOMPOSITION\_TYPE::TOPOLOGY**

A pointer to the topology for this decomposition.  
Definition at line 732 of file types.f90.

**7.176.2.12 INTEGER(INTG) TYPES::DECOMPOSITION\_TYPE::USER\_NUMBER**

The user defined identifier for the domain decomposition. The user number must be unique.  
Definition at line 722 of file types.f90.

## 7.177 TYPES::DECOMPOSITIONS\_TYPE Struct Reference

Contains information on the domain decompositions defined on a mesh.

Collaboration diagram for TYPES::DECOMPOSITIONS\_TYPE:

### Public Attributes

- TYPE(MESH\_TYPE), pointer MESH  
*A pointer to the mesh.*
- INTEGER(INTG) NUMBER\_OF\_DECOMPOSITIONS  
*The number of decompositions defined on the mesh.*
- TYPE(DECOMPOSITION\_PTR\_TYPE), pointer DECOMPOSITIONS  
*DECOMPOSITIONS(decomposition\_idx). The array of pointers to the domain decompositions.*

### 7.177.1 Detailed Description

Contains information on the domain decompositions defined on a mesh.

Definition at line 742 of file types.f90.

### 7.177.2 Member Data Documentation

#### 7.177.2.1 TYPE(DECOMPOSITION\_PTR\_TYPE),pointer TYPES::DECOMPOSITIONS\_- TYPE::DECOMPOSITIONS

DECOMPOSITIONS(decomposition\_idx). The array of pointers to the domain decompositions.

Definition at line 745 of file types.f90.

#### 7.177.2.2 TYPE(MESH\_TYPE),pointer TYPES::DECOMPOSITIONS\_TYPE::MESH

A pointer to the mesh.

Definition at line 743 of file types.f90.

#### 7.177.2.3 INTEGER(INTG) TYPES::DECOMPOSITIONS\_TYPE::NUMBER\_OF\_- DECOMPOSITIONS

The number of decompositions defined on the mesh.

Definition at line 744 of file types.f90.

## 7.178 TYPES::DISTRIBUTED\_MATRIX\_CMISS\_TYPE Struct Reference

Contains information for a [CMISS](#) distributed matrix.

Collaboration diagram for TYPES::DISTRIBUTED\_MATRIX\_CMISS\_TYPE:

### Public Attributes

- TYPE([DISTRIBUTED\\_MATRIX\\_TYPE](#)), pointer [DISTRIBUTED\\_MATRIX](#)  
*A pointer to the distributed matrix.*
- INTEGER(INTG) [BASE\\_TAG\\_NUMBER](#)  
*The base number for the MPI tag numbers that will be used to communicate the distributed matrix data amongst the domains. The base tag number can be thought of as the identification number for the distributed matrix object.*
- TYPE([MATRIX\\_TYPE](#)), pointer [MATRIX](#)  
*A pointer to the matrix to store the rows corresponding to this domain.*

### 7.178.1 Detailed Description

Contains information for a [CMISS](#) distributed matrix.

Definition at line 523 of file types.f90.

### 7.178.2 Member Data Documentation

#### 7.178.2.1 INTEGER(INTG) TYPES::DISTRIBUTED\_MATRIX\_CMISS\_TYPE::BASE\_TAG\_NUMBER

The base number for the MPI tag numbers that will be used to communicate the distributed matrix data amongst the domains. The base tag number can be thought of as the identification number for the distributed matrix object.

Definition at line 525 of file types.f90.

#### 7.178.2.2 TYPE(DISTRIBUTED\_MATRIX\_TYPE),pointer TYPES::DISTRIBUTED\_MATRIX\_CMISS\_TYPE::DISTRIBUTED\_MATRIX

A pointer to the distributed matrix.

Definition at line 524 of file types.f90.

#### 7.178.2.3 TYPE(MATRIX\_TYPE),pointer TYPES::DISTRIBUTED\_MATRIX\_CMISS\_TYPE::MATRIX

A pointer to the matrix to store the rows corresponding to this domain.

Definition at line 526 of file types.f90.

## 7.179 TYPES::DISTRIBUTED\_MATRIX\_PETSC\_TYPE Struct Reference

Contains information for a PETSc distributed matrix.

Collaboration diagram for TYPES::DISTRIBUTED\_MATRIX\_PETSC\_TYPE:

### Public Attributes

- TYPE(DISTRIBUTED\_MATRIX\_TYPE), pointer DISTRIBUTED\_MATRIX  
*A pointer to the distributed matrix.*
- INTEGER(INTG) M  
*The number of local rows in the PETSc matrix.*
- INTEGER(INTG) N  
*The number of local columns in the PETSc matrix.*
- INTEGER(INTG) GLOBAL\_M  
*The number of global rows in the PETSc matrix.*
- INTEGER(INTG) GLOBAL\_N  
*The number of global columns in the PETSc matrix.*
- INTEGER(INTG) STORAGE\_TYPE  
*The storage type (sparsity) of the PETSc matrix.*
- INTEGER(INTG) NUMBER\_NON\_ZEROS  
*The number of non-zeros in the PETSc matrix.*
- INTEGER(INTG) DATA\_SIZE  
*The size of the allocated data in the PETSc matrix.*
- INTEGER(INTG) MAXIMUM\_COLUMN\_INDICES\_PER\_ROW  
*The maximum number of column indices for the rows.*
- INTEGER(INTG), allocatable DIAGONAL\_NUMBER\_NON\_ZEROS  
*DIAGONAL\_NUMBER\_NON\_ZEROS(i). The number of non-zeros in the diagonal part of the the i'th row.*
- INTEGER(INTG), allocatable OFFDIAGONAL\_NUMBER\_NON\_ZEROS  
*OFFDIAGONAL\_NUMBER\_NON\_ZEROS(i). The number of non-zeros in the off diagonal part of the the i'th row.*
- INTEGER(INTG), allocatable ROW\_INDICES  
*ROW\_INDICES(i). The row indices for the matrix.*
- INTEGER(INTG), allocatable COLUMN\_INDICES  
*COLUMN\_INDICES(i). The column indices for the matrix.*
- INTEGER(INTG), allocatable GLOBAL\_ROW\_NUMBERS

*GLOBAL\_ROW\_NUMBERS(i).* The PETSc global row number corresponding to the i'th local row number.

- REAL(DP), allocatable **DATA\_DP**

*DATA\_DP(i).* The real data for the matrix.

- TYPE([PETSC\\_ISLOCALTOGLOBALMAPPING\\_TYPE](#)) **ISLTGMAPPING**

*The local to global mapping for the vector.*

- TYPE([PETSC\\_MAT\\_TYPE](#)) **MATRIX**

*The PETSc matrix.*

### 7.179.1 Detailed Description

Contains information for a PETSc distributed matrix.

Definition at line 530 of file types.f90.

### 7.179.2 Member Data Documentation

#### 7.179.2.1 INTEGER(INTG),allocatable **TYPES::DISTRIBUTED\_MATRIX\_PETSC\_-TYPE::COLUMN\_INDICES**

COLUMN\_INDICES(i). The column indices for the matrix.

Definition at line 543 of file types.f90.

#### 7.179.2.2 REAL(DP),allocatable **TYPES::DISTRIBUTED\_MATRIX\_PETSC\_TYPE::DATA\_DP**

DATA\_DP(i). The real data for the matrix.

Definition at line 545 of file types.f90.

#### 7.179.2.3 INTEGER(INTG) **TYPES::DISTRIBUTED\_MATRIX\_PETSC\_TYPE::DATA\_SIZE**

The size of the allocated data in the PETSc matrix.

Definition at line 538 of file types.f90.

#### 7.179.2.4 INTEGER(INTG),allocatable **TYPES::DISTRIBUTED\_MATRIX\_PETSC\_-TYPE::DIAGONAL\_NUMBER\_NON\_ZEROS**

DIAGONAL\_NUMBER\_NON\_ZEROS(i). The number of non-zeros in the diagonal part of the the i'th row.

Definition at line 540 of file types.f90.

#### 7.179.2.5 TYPE(**DISTRIBUTED\_MATRIX\_TYPE**),pointer **TYPES::DISTRIBUTED\_-MATRIX\_PETSC\_TYPE::DISTRIBUTED\_MATRIX**

A pointer to the distributed matrix.

Definition at line 531 of file types.f90.

#### 7.179.2.6 INTEGER(INTG) TYPES::DISTRIBUTED\_MATRIX\_PETSC\_TYPE::GLOBAL\_M

The number of global rows in the PETSc matrix.

Definition at line 534 of file types.f90.

#### 7.179.2.7 INTEGER(INTG) TYPES::DISTRIBUTED\_MATRIX\_PETSC\_TYPE::GLOBAL\_N

The number of global columns in the PETSc matrix.

Definition at line 535 of file types.f90.

#### 7.179.2.8 INTEGER(INTG),allocatable TYPES::DISTRIBUTED\_MATRIX\_PETSC\_- TYPE::GLOBAL\_ROW\_NUMBERS

GLOBAL\_ROW\_NUMBERS(i). The PETSc global row number corresponding to the i'th local row number.

Definition at line 544 of file types.f90.

#### 7.179.2.9 TYPE(PETSC\_ISLOCALTOGLOBALMAPPING\_TYPE) TYPES::DISTRIBUTED\_MATRIX\_PETSC\_TYPE::ISLTGMAPPING

The local to global mapping for the vector.

Definition at line 546 of file types.f90.

#### 7.179.2.10 INTEGER(INTG) TYPES::DISTRIBUTED\_MATRIX\_PETSC\_TYPE::M

The number of local rows in the PETSc matrix.

Definition at line 532 of file types.f90.

#### 7.179.2.11 TYPE(PETSC\_MAT\_TYPE) TYPES::DISTRIBUTED\_MATRIX\_PETSC\_- TYPE::MATRIX

The PETSc matrix.

Definition at line 547 of file types.f90.

#### 7.179.2.12 INTEGER(INTG) TYPES::DISTRIBUTED\_MATRIX\_PETSC\_TYPE::MAXIMUM\_- COLUMN\_INDICES\_PER\_ROW

The maximum number of column indices for the rows.

Definition at line 539 of file types.f90.

#### 7.179.2.13 INTEGER(INTG) TYPES::DISTRIBUTED\_MATRIX\_PETSC\_TYPE::N

The number of local columns in the PETSc matrix.

Definition at line 533 of file types.f90.

**7.179.2.14 INTEGER(INTG) TYPES::DISTRIBUTED\_MATRIX\_PETSC\_TYPE::NUMBER\_-  
NON\_ZEROS**

The number of non-zeros in the PETSc matrix.

Definition at line 537 of file types.f90.

**7.179.2.15 INTEGER(INTG),allocatable TYPES::DISTRIBUTED\_MATRIX\_PETSC\_-  
TYPE::OFFDIAGONAL\_NUMBER\_NON\_ZEROS**

OFFDIAGONAL\_NUMBER\_NON\_ZEROS(i). The number of non-zeros in the off diagonal part of the  
the i'th row.

Definition at line 541 of file types.f90.

**7.179.2.16 INTEGER(INTG),allocatable TYPES::DISTRIBUTED\_MATRIX\_PETSC\_-  
TYPE::ROW\_INDICES**

ROW\_INDICES(i). The row indices for the matrix.

Definition at line 542 of file types.f90.

**7.179.2.17 INTEGER(INTG) TYPES::DISTRIBUTED\_MATRIX\_PETSC\_TYPE::STORAGE\_-  
TYPE**

The storage type (sparsity) of the PETSc matrix.

Definition at line 536 of file types.f90.

## 7.180 TYPES::DISTRIBUTED\_MATRIX\_TYPE Struct Reference

Contains the information for a matrix that is distributed across a number of domains.

Collaboration diagram for TYPES::DISTRIBUTED\_MATRIX\_TYPE:

### Public Attributes

- LOGICAL [MATRIX\\_FINISHED](#)  
*Is .TRUE. if the distributed matrix has finished being created, .FALSE. if not.*
- INTEGER(INTG) [LIBRARY\\_TYPE](#)  
*The library of the distributed matrix.*
- INTEGER(INTG) [GHOSTING\\_TYPE](#)  
*The ghosting type.*
- TYPE([DOMAIN\\_MAPPING\\_TYPE](#)), pointer [ROW\\_DOMAIN\\_MAPPING](#)  
*The pointer for the domain mapping that identifies how the matrix rows are distributed amongst the domains.*
- TYPE([DOMAIN\\_MAPPING\\_TYPE](#)), pointer [COLUMN\\_DOMAIN\\_MAPPING](#)  
*The pointer for the domain mapping that identifies how the matrix columns are distributed amongst the domains.*
- INTEGER(INTG) [DATA\\_TYPE](#)  
*The type of data for the distributed matrix.*
- TYPE([DISTRIBUTED\\_MATRIX\\_CMISS\\_TYPE](#)), pointer [CMISS](#)  
*A pointer to the [CMISS](#) distributed matrix information.*
- TYPE([DISTRIBUTED\\_MATRIX\\_PETSC\\_TYPE](#)), pointer [PETSC](#)  
*A pointer to the PETSc distributed matrix information.*

### 7.180.1 Detailed Description

Contains the information for a matrix that is distributed across a number of domains.

Definition at line 551 of file types.f90.

### 7.180.2 Member Data Documentation

#### 7.180.2.1 TYPE([DISTRIBUTED\\_MATRIX\\_CMISS\\_TYPE](#)),pointer TYPES::DISTRIBUTED\_MATRIX\_TYPE::CMISS

A pointer to the [CMISS](#) distributed matrix information.

Definition at line 558 of file types.f90.

**7.180.2.2 TYPE(DOMAIN\_MAPPING\_TYPE),pointer TYPES::DISTRIBUTED\_MATRIX\_-  
TYPE::COLUMN\_DOMAIN\_MAPPING**

The pointer for the domain mapping that identifies how the matrix columns are distributed amongst the domains.

Definition at line 556 of file types.f90.

**7.180.2.3 INTEGER(INTG) TYPES::DISTRIBUTED\_MATRIX\_TYPE::DATA\_TYPE**

The type of data for the distributed matrix.

**See also:**

DISTRIBUTED\_MATRIX\_VECTOR\_DataTypes

Definition at line 557 of file types.f90.

**7.180.2.4 INTEGER(INTG) TYPES::DISTRIBUTED\_MATRIX\_TYPE::GHOSTING\_TYPE**

The ghosting type.

**See also:**

DISTRIBUTED\_MATRIX\_VECTOR\_GhostingTypes,DISTRIBUTED\_MATRIX\_VECTOR

Definition at line 554 of file types.f90.

**7.180.2.5 INTEGER(INTG) TYPES::DISTRIBUTED\_MATRIX\_TYPE::LIBRARY\_TYPE**

The library of the distributed matrix.

**See also:**

DISTRIBUTED\_MATRIX\_VECTOR\_LibraryTypes,DISTRIBUTED\_MATRIX\_VECTOR

Definition at line 553 of file types.f90.

**7.180.2.6 LOGICAL TYPES::DISTRIBUTED\_MATRIX\_TYPE::MATRIX\_FINISHED**

Is .TRUE. if the distributed matrix has finished being created, .FALSE. if not.

Definition at line 552 of file types.f90.

**7.180.2.7 TYPE(DISTRIBUTED\_MATRIX\_PETSC\_TYPE),pointer  
TYPES::DISTRIBUTED\_MATRIX\_TYPE::PETSC**

A pointer to the PETSc distributed matrix information.

Definition at line 559 of file types.f90.

**7.180.2.8 TYPE(DOMAIN\_MAPPING\_TYPE),pointer TYPES::DISTRIBUTED\_MATRIX\_-  
TYPE::ROW\_DOMAIN\_MAPPING**

The pointer for the domain mapping that identifies how the matrix rows are distributed amongst the domains.

Definition at line 555 of file types.f90.

## 7.181 TYPES::DISTRIBUTED\_VECTOR\_CMISS\_TYPE Struct Reference

Contains information for a [CMISS](#) distributed vector.

Collaboration diagram for TYPES::DISTRIBUTED\_VECTOR\_CMISS\_TYPE:

### Public Attributes

- TYPE([DISTRIBUTED\\_VECTOR\\_TYPE](#)), pointer [DISTRIBUTED\\_VECTOR](#)  
*A pointer to the distributed vector.*
- INTEGER(INTG) [BASE\\_TAG\\_NUMBER](#)  
*The base number for the MPI tag numbers that will be used to communicate the distributed vector data amongst the domains. The base tag number can be thought of as the identification number for the distributed vector object.*
- INTEGER(INTG) [N](#)  
*The size of the distributed vector.*
- INTEGER(INTG) [DATA\\_SIZE](#)  
*The size of the distributed vector that is held locally by the domain.*
- INTEGER(INTG), allocatable [DATA\\_INTG](#)  
*DATA\_INTG(i). The integer data for an integer distributed vector. The i'th component contains the data for the i'th local number of distributed vector data on the domain.*
- REAL(DP), allocatable [DATA\\_DP](#)  
*DATA\_DP(i). The real data for a double precision real distributed vector. The i'th component contains the data for the i'th local number of distributed vector data on the domain.*
- REAL(SP), allocatable [DATA\\_SP](#)  
*DATA\_SP(i). The real data for a single precision real distributed vector. The i'th component contains the data for the i'th local number of distributed vector data on the domain.*
- LOGICAL, allocatable [DATA\\_L](#)  
*DATA\_L(i). The logical data for a logical distributed vector. The i'th component contains the data for the i'th local number of distributed vector data on the domain.*
- TYPE([DISTRIBUTED\\_VECTOR\\_TRANSFER\\_TYPE](#)), allocatable [TRANSFERS](#)  
*TRANSFERS(adjacent\_domain\_idx). The transfer information for the adjacent\_domain\_idx'th adjacent domain to this domain.*

### 7.181.1 Detailed Description

Contains information for a [CMISS](#) distributed vector.

Definition at line 488 of file types.f90.

## 7.181.2 Member Data Documentation

### 7.181.2.1 INTEGER(INTG) TYPES::DISTRIBUTED\_VECTOR\_CMISS\_TYPE::BASE\_TAG - NUMBER

The base number for the MPI tag numbers that will be used to communicate the distributed vector data amongst the domains. The base tag number can be thought of as the identification number for the distributed vector object.

Definition at line 490 of file types.f90.

### 7.181.2.2 REAL(DP),allocatable TYPES::DISTRIBUTED\_VECTOR\_CMISS\_TYPE::DATA\_DP

DATA\_DP(i). The real data for a double precision real distributed vector. The i'th component contains the data for the i'th local number of distributed vector data on the domain.

Definition at line 494 of file types.f90.

### 7.181.2.3 INTEGER(INTG),allocatable TYPES::DISTRIBUTED\_VECTOR\_CMISS\_TYPE::DATA\_INTG

DATA\_INTG(i). The integer data for an integer distributed vector. The i'th component contains the data for the i'th local number of distributed vector data on the domain.

Definition at line 493 of file types.f90.

### 7.181.2.4 LOGICAL,allocatable TYPES::DISTRIBUTED\_VECTOR\_CMISS\_TYPE::DATA\_L

DATA\_L(i). The logical data for a logical distributed vector. The i'th component contains the data for the i'th local number of distributed vector data on the domain.

Definition at line 496 of file types.f90.

### 7.181.2.5 INTEGER(INTG) TYPES::DISTRIBUTED\_VECTOR\_CMISS\_TYPE::DATA\_SIZE

The size of the distributed vector that is held locally by the domain.

Definition at line 492 of file types.f90.

### 7.181.2.6 REAL(SP),allocatable TYPES::DISTRIBUTED\_VECTOR\_CMISS\_TYPE::DATA\_SP

DATA\_SP(i). The real data for a single precision real distributed vector. The i'th component contains the data for the i'th local number of distributed vector data on the domain.

Definition at line 495 of file types.f90.

### 7.181.2.7 TYPE(DISTRIBUTED\_VECTOR\_TYPE),pointer TYPES::DISTRIBUTED\_VECTOR\_CMISS\_TYPE::DISTRIBUTED\_VECTOR

A pointer to the distributed vector.

Definition at line 489 of file types.f90.

**7.181.2.8 INTEGER(INTG) TYPES::DISTRIBUTED\_VECTOR\_CMISS\_TYPE::N**

The size of the distributed vector.

Definition at line 491 of file types.f90.

**7.181.2.9 TYPE(DISTRIBUTED\_VECTOR\_TRANSFER\_TYPE),allocatable  
TYPES::DISTRIBUTED\_VECTOR\_CMISS\_TYPE::TRANSFERS**

TRANSFERS(adjacent\_domain\_idx). The transfer information for the adjacent\_domain\_idx'th adjacent domain to this domain.

Definition at line 497 of file types.f90.

## 7.182 TYPES::DISTRIBUTED\_VECTOR\_PETSC\_TYPE Struct Reference

Contains information for a PETSc distributed vector.

Collaboration diagram for TYPES::DISTRIBUTED\_VECTOR\_PETSC\_TYPE:

### Public Attributes

- TYPE(DISTRIBUTED\_VECTOR\_TYPE), pointer DISTRIBUTED\_VECTOR  
*A pointer to the distributed vector.*
- INTEGER(INTG) N  
*The number of local components in the vector.*
- INTEGER(INTG) GLOBAL\_N  
*The number of global components in the vector.*
- INTEGER(INTG) DATA\_SIZE  
*The size of the distributed vector that is held locally by the domain.*
- INTEGER(INTG), allocatable GLOBAL\_NUMBERS  
*GLOBAL\_NUMBERS(i). The PETSc global number corresponding to the i'th local number.*
- TYPE(PETSC\_ISLOCALTOGLOBALMAPPING\_TYPE) ISLTGMAPPING  
*The local to global mapping for the vector.*
- TYPE(PETSC\_VEC\_TYPE) VECTOR  
*The PETSc vector.*

### 7.182.1 Detailed Description

Contains information for a PETSc distributed vector.

Definition at line 501 of file types.f90.

### 7.182.2 Member Data Documentation

#### 7.182.2.1 INTEGER(INTG) TYPES::DISTRIBUTED\_VECTOR\_PETSC\_TYPE::DATA\_SIZE

The size of the distributed vector that is held locally by the domain.

Definition at line 505 of file types.f90.

#### 7.182.2.2 TYPE(DISTRIBUTED\_VECTOR\_TYPE),pointer TYPES::DISTRIBUTED\_VECTOR\_PETSC\_TYPE::DISTRIBUTED\_VECTOR

A pointer to the distributed vector.

Definition at line 502 of file types.f90.

**7.182.2.3 INTEGER(INTG) TYPES::DISTRIBUTED\_VECTOR\_PETSC\_TYPE::GLOBAL\_N**

The number of global components in the vector.

Definition at line 504 of file types.f90.

**7.182.2.4 INTEGER(INTG),allocatable TYPES::DISTRIBUTED\_VECTOR\_PETSC\_-  
TYPE::GLOBAL\_NUMBERS**

GLOBAL\_NUMBERS(i). The PETSc global number corresponding to the i'th local number.

Definition at line 506 of file types.f90.

**7.182.2.5 TYPE(PETSC\_ISLOCALTOGLOBALMAPPING\_TYPE)  
TYPES::DISTRIBUTED\_VECTOR\_PETSC\_TYPE::ISLTGMAPPING**

The local to global mapping for the vector.

Definition at line 507 of file types.f90.

**7.182.2.6 INTEGER(INTG) TYPES::DISTRIBUTED\_VECTOR\_PETSC\_TYPE::N**

The number of local components in the vector.

Definition at line 503 of file types.f90.

**7.182.2.7 TYPE(PETSC\_VEC\_TYPE) TYPES::DISTRIBUTED\_VECTOR\_PETSC\_-  
TYPE::VECTOR**

The PETSc vector.

Definition at line 508 of file types.f90.

## 7.183 TYPES::DISTRIBUTED\_VECTOR\_TRANSFER\_TYPE Struct Reference

Contains the information for an adjacent domain for transferring the ghost data of a distributed vector to/from the current domain.

Collaboration diagram for TYPES::DISTRIBUTED\_VECTOR\_TRANSFER\_TYPE:

### Public Attributes

- TYPE(DISTRIBUTED\_VECTOR\_CMISS\_TYPE), pointer CMISS\_VECTOR  
*The pointer to the CMISS distributed vector object for this transfer information.*
- INTEGER(INTG) DATA\_TYPE  
*The data type of the distributed vector. This is "inherited" from the distributed vector.*
- INTEGER(INTG) SEND\_BUFFER\_SIZE  
*The size of the buffer to send distributed vector data from the current domain to the adjacent domain.*
- INTEGER(INTG) RECEIVE\_BUFFER\_SIZE  
*The size of the buffer to receive distributed vector data from the adjacent domain to the current domain.*
- INTEGER(INTG) SEND\_TAG\_NUMBER  
*The MPI tag number for the data sending from the current domain to the adjacent domain. It is calculated as an offset from the base tag number of the distributed vector.*
- INTEGER(INTG) RECEIVE\_TAG\_NUMBER  
*The MPI tag number for the data receiving from the adjacent domain to the current domain. It is calculated as an offset from the base tag number of the distributed vector.*
- INTEGER(INTG) MPI\_SEND\_REQUEST  
*The MPI request pointer for sending data from the current domain to the adjacent domain.*
- INTEGER(INTG) MPI\_RECEIVE\_REQUEST  
*The MPI request pointer for sending data from the adjacent domain to the current domain.*
- INTEGER(INTG), allocatable SEND\_BUFFER\_INTG  
*The integer buffer for sending the distributed integer vector data from the current domain to the adjacent domain.*
- REAL(DP), allocatable SEND\_BUFFER\_DP  
*The double precision real buffer for sending the distributed real vector data from the current domain to the adjacent domain.*
- REAL(SP), allocatable SEND\_BUFFER\_SP  
*The single precision real buffer for sending the distributed real vector data from the current domain to the adjacent domain.*
- LOGICAL, allocatable SEND\_BUFFER\_L  
*The logical buffer for sending the distributed logical vector data from the current domain to the adjacent domain.*

- INTEGER(INTG), allocatable **RECEIVE\_BUFFER\_INTG**

*The integer buffer for receiving the distributed integer vector data from the adjacent domain to the current domain.*

- REAL(DP), allocatable **RECEIVE\_BUFFER\_DP**

*The double precision real buffer for receiving the distributed real vector data from the adjacent domain to the current domain.*

- REAL(SP), allocatable **RECEIVE\_BUFFER\_SP**

*The single precision real buffer for receiving the distributed real vector data from the adjacent domain to the current domain.*

- LOGICAL, allocatable **RECEIVE\_BUFFER\_L**

*The logical buffer for receiving the distributed logical vector data from the adjacent domain to the current domain.*

### 7.183.1 Detailed Description

Contains the information for an adjacent domain for transferring the ghost data of a distributed vector to/from the current domain.

Definition at line 468 of file types.f90.

### 7.183.2 Member Data Documentation

#### 7.183.2.1 **TYPE(DISTRIBUTED\_VECTOR\_CMISS\_TYPE),pointer TYPES::DISTRIBUTED\_VECTOR\_TRANSFER\_TYPE::CMISS\_VECTOR**

The pointer to the **CMISS** distributed vector object for this transfer information.

Definition at line 469 of file types.f90.

#### 7.183.2.2 **INTEGER(INTG) TYPES::DISTRIBUTED\_VECTOR\_TRANSFER\_TYPE::DATA\_TYPE**

The data type of the distributed vector. This is "inherited" from the distributed vector.

Definition at line 470 of file types.f90.

#### 7.183.2.3 **INTEGER(INTG) TYPES::DISTRIBUTED\_VECTOR\_TRANSFER\_TYPE::MPI\_RECEIVE\_REQUEST**

The MPI request pointer for sending data from the adjacent domain to the current domain.

Definition at line 476 of file types.f90.

#### 7.183.2.4 **INTEGER(INTG) TYPES::DISTRIBUTED\_VECTOR\_TRANSFER\_TYPE::MPI\_SEND\_REQUEST**

The MPI request pointer for sending data from the current domain to the adjacent domain.

Definition at line 475 of file types.f90.

#### **7.183.2.5 REAL(DP),allocatable TYPES::DISTRIBUTED\_VECTOR\_TRANSFER\_- TYPE::RECEIVE\_BUFFER\_DP**

The double precision real buffer for receiving the distributed real vector data from the adjacent domain to the current domain.

Definition at line 482 of file types.f90.

#### **7.183.2.6 INTEGER(INTG),allocatable TYPES::DISTRIBUTED\_VECTOR\_TRANSFER\_- TYPE::RECEIVE\_BUFFER\_INTG**

The integer buffer for receiving the distributed integer vector data from the adjacent domain to the current domain.

Definition at line 481 of file types.f90.

#### **7.183.2.7 LOGICAL,allocatable TYPES::DISTRIBUTED\_VECTOR\_TRANSFER\_- TYPE::RECEIVE\_BUFFER\_L**

The logical buffer for receiving the distributed logical vector data from the adjacent domain to the current domain.

Definition at line 484 of file types.f90.

#### **7.183.2.8 INTEGER(INTG) TYPES::DISTRIBUTED\_VECTOR\_TRANSFER\_- TYPE::RECEIVE\_BUFFER\_SIZE**

The size of the buffer to receive distributed vector data from the adjacent domain to the current domain.

Definition at line 472 of file types.f90.

#### **7.183.2.9 REAL(SP),allocatable TYPES::DISTRIBUTED\_VECTOR\_TRANSFER\_- TYPE::RECEIVE\_BUFFER\_SP**

The single precision real buffer for receiving the distributed real vector data from the adjacent domain to the current domain.

Definition at line 483 of file types.f90.

#### **7.183.2.10 INTEGER(INTG) TYPES::DISTRIBUTED\_VECTOR\_TRANSFER\_- TYPE::RECEIVE\_TAG\_NUMBER**

The MPI tag number for the data receiving from the adjacent domain to the current domain. It is calculated as an offset from the base tag number of the distributed vector.

Definition at line 474 of file types.f90.

**7.183.2.11 REAL(DP),allocatable TYPES::DISTRIBUTED\_VECTOR\_TRANSFER\_-  
TYPE::SEND\_BUFFER\_DP**

The double precision real buffer for sending the distributed real vector data from the current domain to the adjacent domain.

Definition at line 478 of file types.f90.

**7.183.2.12 INTEGER(INTG),allocatable TYPES::DISTRIBUTED\_VECTOR\_TRANSFER\_-  
TYPE::SEND\_BUFFER\_INTG**

The integer buffer for sending the distributed integer vector data from the current domain to the adjacent domain.

Definition at line 477 of file types.f90.

**7.183.2.13 LOGICAL,allocatable TYPES::DISTRIBUTED\_VECTOR\_TRANSFER\_-  
TYPE::SEND\_BUFFER\_L**

The logical buffer for sending the distributed logical vector data from the current domain to the adjacent domain.

Definition at line 480 of file types.f90.

**7.183.2.14 INTEGER(INTG) TYPES::DISTRIBUTED\_VECTOR\_TRANSFER\_TYPE::SEND\_-  
BUFFER\_SIZE**

The size of the buffer to send distributed vector data from the current domain to the adjacent domain.

Definition at line 471 of file types.f90.

**7.183.2.15 REAL(SP),allocatable TYPES::DISTRIBUTED\_VECTOR\_TRANSFER\_-  
TYPE::SEND\_BUFFER\_SP**

The single precision real buffer for sending the distributed real vector data from the current domain to the adjacent domain.

Definition at line 479 of file types.f90.

**7.183.2.16 INTEGER(INTG) TYPES::DISTRIBUTED\_VECTOR\_TRANSFER\_TYPE::SEND\_-  
TAG\_NUMBER**

The MPI tag number for the data sending from the current domain to the adjacent domain. It is calculated as an offset from the base tag number of the distributed vector.

Definition at line 473 of file types.f90.

## 7.184 TYPES::DISTRIBUTED\_VECTOR\_TYPE Struct Reference

Contains the information for a vector that is distributed across a number of domains.

Collaboration diagram for TYPES::DISTRIBUTED\_VECTOR\_TYPE:

### Public Attributes

- LOGICAL [VECTOR\\_FINISHED](#)  
*!<Is .TRUE. if the distributed vector has finished being created, .FALSE. if not.*
- INTEGER(INTG) [LIBRARY\\_TYPE](#)  
*The format of the distributed vector.*
- INTEGER(INTG) [GHOSTING\\_TYPE](#)  
*The ghosting type.*
- TYPE([DOMAIN\\_MAPPING\\_TYPE](#)), pointer [DOMAIN\\_MAPPING](#)  
*The pointer for the domain mapping that identifies how the vector is distributed amongst the domains.*
- INTEGER(INTG) [DATA\\_TYPE](#)  
*The type of data for the distributed vector.*
- TYPE([DISTRIBUTED\\_VECTOR\\_CMISS\\_TYPE](#)), pointer [CMISS](#)  
*A pointer to the [CMISS](#) distributed vector information.*
- TYPE([DISTRIBUTED\\_VECTOR\\_PETSC\\_TYPE](#)), pointer [PETSC](#)  
*A pointer to the PETSc distributed vector information.*

### 7.184.1 Detailed Description

Contains the information for a vector that is distributed across a number of domains.

Definition at line 512 of file types.f90.

### 7.184.2 Member Data Documentation

#### 7.184.2.1 TYPE([DISTRIBUTED\\_VECTOR\\_CMISS\\_TYPE](#)),pointer TYPES::DISTRIBUTED\_VECTOR\_TYPE::CMISS

A pointer to the [CMISS](#) distributed vector information.

Definition at line 518 of file types.f90.

#### 7.184.2.2 INTEGER(INTG) TYPES::DISTRIBUTED\_VECTOR\_TYPE::DATA\_TYPE

The type of data for the distributed vector.

**See also:**

[DISTRIBUTED\\_MATRIX\\_VECTOR\\_DataTypes](#)

Definition at line 517 of file types.f90.

**7.184.2.3 TYPE(DOMAIN\_MAPPING\_TYPE),pointer TYPES::DISTRIBUTED\_VECTOR\_-  
TYPE::DOMAIN\_MAPPING**

The pointer for the domain mapping that identifies how the vector is distributed amongst the domains.

Definition at line 516 of file types.f90.

**7.184.2.4 INTEGER(INTG) TYPES::DISTRIBUTED\_VECTOR\_TYPE::GHOSTING\_TYPE**

The ghosting type.

**See also:**

[DISTRIBUTED\\_MATRIX\\_VECTOR\\_GhostingTypes](#),[DISTRIBUTED\\_MATRIX\\_VECTOR](#)

Definition at line 515 of file types.f90.

**7.184.2.5 INTEGER(INTG) TYPES::DISTRIBUTED\_VECTOR\_TYPE::LIBRARY\_TYPE**

The format of the distributed vector.

**See also:**

[DISTRIBUTED\\_MATRIX\\_VECTOR\\_LibraryTypes](#),[DISTRIBUTED\\_MATRIX\\_VECTOR](#)

Definition at line 514 of file types.f90.

**7.184.2.6 TYPE(DISTRIBUTED\_VECTOR\_PETSC\_TYPE),pointer  
TYPES::DISTRIBUTED\_VECTOR\_TYPE::PETSC**

A pointer to the PETSc distributed vector information.

Definition at line 519 of file types.f90.

**7.184.2.7 LOGICAL TYPES::DISTRIBUTED\_VECTOR\_TYPE::VECTOR\_FINISHED**

!<Is .TRUE. if the distributed vector has finished being created, .FALSE. if not.

Definition at line 513 of file types.f90.

## 7.185 TYPES::DOMAIN\_ADJACENT\_DOMAIN\_TYPE Struct Reference

Contains the information on an adjacent domain to a domain in a domain mapping.

### Public Attributes

- INTEGER(INTG) [DOMAIN\\_NUMBER](#)

*The number of the domain that is adjacent to a domain in a mapping.*

- INTEGER(INTG) [NUMBER\\_OF\\_SEND\\_GHOSTS](#)

*The number of ghost locals in the current domain that are to be sent to this domain.*

- INTEGER(INTG) [NUMBER\\_OF\\_RECEIVE\\_GHOSTS](#)

*The number of ghost locals in the current domain that are to be received from this domain.*

- INTEGER(INTG), allocatable [LOCAL\\_GHOST\\_SEND\\_INDICES](#)

*LOCAL\_GHOST\_SEND\_INDICES(i). The local numbers of the ghosts in the current domain that are to be sent to this domain.*

- INTEGER(INTG), allocatable [LOCAL\\_GHOST\\_RECEIVE\\_INDICES](#)

*LOCAL\_GHOST\_RECEIVE\_INDICES(i). The local numbers of the ghosts in the current domain that are to be received from this domain.*

### 7.185.1 Detailed Description

Contains the information on an adjacent domain to a domain in a domain mapping.

Definition at line 609 of file types.f90.

### 7.185.2 Member Data Documentation

#### 7.185.2.1 INTEGER(INTG) TYPES::DOMAIN\_ADJACENT\_DOMAIN\_TYPE::DOMAIN\_NUMBER

The number of the domain that is adjacent to a domain in a mapping.

Definition at line 610 of file types.f90.

#### 7.185.2.2 INTEGER(INTG),allocatable TYPES::DOMAIN\_ADJACENT\_DOMAIN\_TYPE::LOCAL\_GHOST\_RECEIVE\_INDICES

*LOCAL\_GHOST\_RECEIVE\_INDICES(i). The local numbers of the ghosts in the current domain that are to be received from this domain.*

Definition at line 614 of file types.f90.

**7.185.2.3 INTEGER(INTG),allocatable TYPES::DOMAIN\_ADJACENT\_DOMAIN\_-  
TYPE::LOCAL\_GHOST\_SEND\_INDICES**

LOCAL\_GHOST\_SEND\_INDICES(i). The local numbers of the ghosts in the current domain that are to be sent to this domain.

Definition at line 613 of file types.f90.

**7.185.2.4 INTEGER(INTG) TYPES::DOMAIN\_ADJACENT\_DOMAIN\_TYPE::NUMBER\_-  
OF\_RECEIVE\_GHOSTS**

The number of ghost locals in the current domain that are to be received from this domain.

Definition at line 612 of file types.f90.

**7.185.2.5 INTEGER(INTG) TYPES::DOMAIN\_ADJACENT\_DOMAIN\_TYPE::NUMBER\_-  
OF\_SEND\_GHOSTS**

The number of ghost locals in the current domain that are to be sent to this domain.

Definition at line 611 of file types.f90.

## 7.186 TYPES::DOMAIN\_DOFS\_TYPE Struct Reference

Contains information on the degrees-of-freedom (dofs) for a domain.

Collaboration diagram for TYPES::DOMAIN\_DOFS\_TYPE:

### Public Attributes

- TYPE(DOMAIN\_TYPE), pointer DOMAIN

*A pointer to the domain.*

- INTEGER(INTG) NUMBER\_OF\_DOFS

*The number of degrees-of-freedom (excluding ghost dofs) in the domain.*

- INTEGER(INTG) TOTAL\_NUMBER\_OF\_DOFS

*The total number of degrees-of-freedom (including ghost dofs) in the domain.*

- INTEGER(INTG), allocatable DOF\_INDEX

*DOF\_INDEX(i,ny). The index for the ny'th degree-of-freedom. When i=1 DOF\_INDEX will give the global derivative number (nk) associated with the dof. When i=2 DOF\_INDEX will give the local node number (np) associated with the dof.*

### 7.186.1 Detailed Description

Contains information on the degrees-of-freedom (dofs) for a domain.

Definition at line 361 of file types.f90.

### 7.186.2 Member Data Documentation

#### 7.186.2.1 INTEGER(INTG),allocatable TYPES::DOMAIN\_DOFS\_TYPE::DOF\_INDEX

DOF\_INDEX(i,ny). The index for the ny'th degree-of-freedom. When i=1 DOF\_INDEX will give the global derivative number (nk) associated with the dof. When i=2 DOF\_INDEX will give the local node number (np) associated with the dof.

Definition at line 365 of file types.f90.

#### 7.186.2.2 TYPE(DOMAIN\_TYPE),pointer TYPES::DOMAIN\_DOFS\_TYPE::DOMAIN

A pointer to the domain.

Definition at line 362 of file types.f90.

#### 7.186.2.3 INTEGER(INTG) TYPES::DOMAIN\_DOFS\_TYPE::NUMBER\_OF\_DOFS

The number of degrees-of-freedom (excluding ghost dofs) in the domain.

Definition at line 363 of file types.f90.

**7.186.2.4 INTEGER(INTG) TYPES::DOMAIN\_DOFS\_TYPE::TOTAL\_NUMBER\_OF\_DOFS**

The total number of degrees-of-freedom (including ghost dofs) in the domain.

Definition at line 364 of file types.f90.

## 7.187 TYPES::DOMAIN\_ELEMENT\_TYPE Struct Reference

Contains the information for an element in a domain.

Collaboration diagram for TYPES::DOMAIN\_ELEMENT\_TYPE:

### Public Attributes

- INTEGER(INTG) **NUMBER**  
*The local element number in the domain.*
- TYPE(BASIS\_TYPE), pointer **BASIS**  
*A pointer to the basis function for the element.*
- INTEGER(INTG), allocatable **ELEMENT\_NODES**  
*ELEMENT\_NODES(nn). The local node number in the domain of the nn'th local node in the element. Old CMISS name NPNE(nn,nbf,ne).*
- INTEGER(INTG), allocatable **ELEMENT\_DERIVATIVES**  
*ELEMENT\_DERIVATIVES(nk,nn). The global derivative number of the local derivative nk for the local node nn in the element. Old CMISS name NKJE(nk,nn,nj,ne).*

### 7.187.1 Detailed Description

Contains the information for an element in a domain.

Definition at line 411 of file types.f90.

### 7.187.2 Member Data Documentation

#### 7.187.2.1 TYPE(BASIS\_TYPE),pointer TYPES::DOMAIN\_ELEMENT\_TYPE::BASIS

A pointer to the basis function for the element.

Definition at line 413 of file types.f90.

#### 7.187.2.2 INTEGER(INTG),allocatable TYPES::DOMAIN\_ELEMENT\_TYPE::ELEMENT\_DERIVATIVES

ELEMENT\_DERIVATIVES(nk,nn). The global derivative number of the local derivative nk for the local node nn in the element. Old CMISS name NKJE(nk,nn,nj,ne).

Definition at line 415 of file types.f90.

#### 7.187.2.3 INTEGER(INTG),allocatable TYPES::DOMAIN\_ELEMENT\_TYPE::ELEMENT\_NODES

ELEMENT\_NODES(nn). The local node number in the domain of the nn'th local node in the element. Old CMISS name NPNE(nn,nbf,ne).

Definition at line 414 of file types.f90.

**7.187.2.4 INTEGER(INTG) TYPES::DOMAIN\_ELEMENT\_TYPE::NUMBER**

The local element number in the domain.

Definition at line 412 of file types.f90.

## 7.188 TYPES::DOMAIN\_ELEMENTS\_TYPE Struct Reference

Contains the topology information for the elements of a domain.

Collaboration diagram for TYPES::DOMAIN\_ELEMENTS\_TYPE:

### Public Attributes

- TYPE(DOMAIN\_TYPE), pointer DOMAIN  
*The pointer to the domain for this elements topology information.*
- INTEGER(INTG) NUMBER\_OF\_ELEMENTS  
*The number of elements (excluding ghost elements) in this domain topology.*
- INTEGER(INTG) TOTAL\_NUMBER\_OF\_ELEMENTS  
*The total number of elements (including ghost elements) in this domain topology.*
- TYPE(DOMAIN\_ELEMENT\_TYPE), pointer ELEMENTS  
*ELEMENTS(ne). The pointer to the array of topology information for the elements of this domain. ELEMENTS(ne) contains the topological information for the ne'th local elements of the domain.*
- INTEGER(INTG) MAXIMUM\_NUMBER\_OF\_ELEMENT\_PARAMETERS  
*The maximum number of element parameters (ns) for all the elements in the domain.*

### 7.188.1 Detailed Description

Contains the topology information for the elements of a domain.

Definition at line 419 of file types.f90.

### 7.188.2 Member Data Documentation

#### 7.188.2.1 TYPE(DOMAIN\_TYPE),pointer TYPES::DOMAIN\_ELEMENTS\_TYPE::DOMAIN

The pointer to the domain for this elements topology information.

Definition at line 420 of file types.f90.

#### 7.188.2.2 TYPE(DOMAIN\_ELEMENT\_TYPE),pointer TYPES::DOMAIN\_ELEMENTS\_TYPE::ELEMENTS

ELEMENTS(ne). The pointer to the array of topology information for the elements of this domain. ELEMENTS(ne) contains the topological information for the ne'th local elements of the domain.

#### **Todo**

Change this to allocatable???

Definition at line 423 of file types.f90.

**7.188.2.3 INTEGER(INTG) TYPES::DOMAIN\_ELEMENTS\_TYPE::MAXIMUM\_NUMBER\_OF\_ELEMENT\_PARAMETERS**

The maximum number of element parameters (ns) for all the elements in the domain.

Definition at line 424 of file types.f90.

**7.188.2.4 INTEGER(INTG) TYPES::DOMAIN\_ELEMENTS\_TYPE::NUMBER\_OF\_ELEMENTS**

The number of elements (excluding ghost elements) in this domain topology.

Definition at line 421 of file types.f90.

**7.188.2.5 INTEGER(INTG) TYPES::DOMAIN\_ELEMENTS\_TYPE::TOTAL\_NUMBER\_OF\_ELEMENTS**

The total number of elements (including ghost elements) in this domain topology.

Definition at line 422 of file types.f90.

## 7.189 TYPES::DOMAIN\_FACE\_PTR\_TYPE Struct Reference

A buffer type to allow for an array of pointers to a [FIELD\\_VARIABLE\\_TYPE](#).

Collaboration diagram for TYPES::DOMAIN\_FACE\_PTR\_TYPE:

### Public Attributes

- TYPE([DOMAIN\\_FACE\\_TYPE](#)), pointer PTR

*The pointer to the domain face.*

### 7.189.1 Detailed Description

A buffer type to allow for an array of pointers to a [FIELD\\_VARIABLE\\_TYPE](#).

Definition at line 399 of file types.f90.

### 7.189.2 Member Data Documentation

#### 7.189.2.1 TYPE(DOMAIN\_FACE\_TYPE),pointer TYPES::DOMAIN\_FACE\_PTR\_TYPE::PTR

The pointer to the domain face.

Definition at line 400 of file types.f90.

## 7.190 TYPES::DOMAIN\_FACE\_TYPE Struct Reference

Contains the information for a face in a domain.

Collaboration diagram for TYPES::DOMAIN\_FACE\_TYPE:

### Public Attributes

- INTEGER(INTG) **NUMBER**  
*The face number in the domain.*
- INTEGER(INTG) **XI\_DIRECTION1**  
*The first xi direction of the face.*
- INTEGER(INTG) **XI\_DIRECTION2**  
*The second xi direction of the face.*
- TYPE([BASIS\\_TYPE](#)), pointer **BASIS**  
*A pointer to the basis function for the face.*
- INTEGER(INTG), allocatable **NODES\_IN\_FACE**  
*NODES\_IN\_FACE(nn). The local node number in the domain of the nn'th local node in the face. Old CMISS name NPNF(nn,nbf).*
- INTEGER(INTG), allocatable **DERIVATIVES\_IN\_FACE**  
*DERIVATIVES\_IN\_FACE(nk,nn). The global derivative number of the local derivative nk for the local node nn in the face.*

### 7.190.1 Detailed Description

Contains the information for a face in a domain.

Definition at line 389 of file types.f90.

### 7.190.2 Member Data Documentation

#### 7.190.2.1 TYPE([BASIS\\_TYPE](#)),pointer TYPES::DOMAIN\_FACE\_TYPE::**BASIS**

A pointer to the basis function for the face.

Definition at line 393 of file types.f90.

#### 7.190.2.2 INTEGER(INTG),allocatable TYPES::DOMAIN\_FACE\_TYPE::**DERIVATIVES\_IN\_FACE**

DERIVATIVES\_IN\_FACE(nk,nn). The global derivative number of the local derivative nk for the local node nn in the face.

Definition at line 395 of file types.f90.

**7.190.2.3 INTEGER(INTG),allocatable TYPES::DOMAIN\_FACE\_TYPE::NODES\_IN\_FACE**

NODES\_IN\_FACE(nn). The local node number in the domain of the nn'th local node in the face. Old CMISS name NPNF(nn,nbf).

Definition at line 394 of file types.f90.

**7.190.2.4 INTEGER(INTG) TYPES::DOMAIN\_FACE\_TYPE::NUMBER**

The face number in the domain.

Definition at line 390 of file types.f90.

**7.190.2.5 INTEGER(INTG) TYPES::DOMAIN\_FACE\_TYPE::XI\_DIRECTION1**

The first xi direction of the face.

**Todo**

move this to the decomposition face type

Definition at line 391 of file types.f90.

**7.190.2.6 INTEGER(INTG) TYPES::DOMAIN\_FACE\_TYPE::XI\_DIRECTION2**

The second xi direction of the face.

**Todo**

move this to the decomposition face type

Definition at line 392 of file types.f90.

## 7.191 TYPES::DOMAIN\_FACES\_TYPE Struct Reference

Contains the topology information for the faces of a domain.

Collaboration diagram for TYPES::DOMAIN\_FACES\_TYPE:

### Public Attributes

- TYPE(DOMAIN\_TYPE), pointer DOMAIN  
*The pointer to the domain for this faces topology information.*
- INTEGER(INTG) NUMBER\_OF\_FACES  
*The number of faces in this domain topology.*
- TYPE(DOMAIN\_FACE\_TYPE), allocatable FACES  
*FACES(nf). The pointer to the array of topology information for the faces of this domain. FACES(nf) contains the topological information for the nf'th local face of the domain.*

### 7.191.1 Detailed Description

Contains the topology information for the faces of a domain.

Definition at line 404 of file types.f90.

### 7.191.2 Member Data Documentation

#### 7.191.2.1 TYPE(DOMAIN\_TYPE),pointer TYPES::DOMAIN\_FACES\_TYPE::DOMAIN

The pointer to the domain for this faces topology information.

Definition at line 405 of file types.f90.

#### 7.191.2.2 TYPE(DOMAIN\_FACE\_TYPE),allocatable TYPES::DOMAIN\_FACES\_TYPE::FACES

FACES(nf). The pointer to the array of topology information for the faces of this domain. FACES(nf) contains the topological information for the nf'th local face of the domain.

Definition at line 407 of file types.f90.

#### 7.191.2.3 INTEGER(INTG) TYPES::DOMAIN\_FACES\_TYPE::NUMBER\_OF\_FACES

The number of faces in this domain topology.

Definition at line 406 of file types.f90.

## 7.192 TYPES::DOMAIN\_GLOBAL\_MAPPING\_TYPE Struct Reference

Contains the local information for a global mapping number for a domain mapping.

### Public Attributes

- INTEGER(INTG) [NUMBER\\_OF\\_DOMAINS](#)

*The number of domains that the global number is mapped to a local number in.*

- INTEGER(INTG), allocatable [LOCAL\\_NUMBER](#)

*LOCAL\_NUMBER(domain\_idx). The mapped local number for the domain\_idx'th domain for the global number.*

- INTEGER(INTG), allocatable [DOMAIN\\_NUMBER](#)

*DOMAIN\_NUMBER(domain\_idx). The domain number for the domain\_idx'th domain for which the global number is mapped to a local number.*

- INTEGER(INTG), allocatable [LOCAL\\_TYPE](#)

*LOCAL\_TYPE(domain\_idx). The type of local for the domain\_idx'th domain for which the global number is mapped to a local number. The types depend on whether the mapped local number in the domain\_idx'th domain is an internal, boundary or ghost local number.*

### 7.192.1 Detailed Description

Contains the local information for a global mapping number for a domain mapping.

Definition at line 618 of file types.f90.

### 7.192.2 Member Data Documentation

#### 7.192.2.1 INTEGER(INTG),allocatable TYPES::DOMAIN\_GLOBAL\_MAPPING\_- TYPE::DOMAIN\_NUMBER

DOMAIN\_NUMBER(domain\_idx). The domain number for the domain\_idx'th domain for which the global number is mapped to a local number.

Definition at line 621 of file types.f90.

#### 7.192.2.2 INTEGER(INTG),allocatable TYPES::DOMAIN\_GLOBAL\_MAPPING\_- TYPE::LOCAL\_NUMBER

LOCAL\_NUMBER(domain\_idx). The mapped local number for the domain\_idx'th domain for the global number.

Definition at line 620 of file types.f90.

**7.192.2.3 INTEGER(INTG),allocatable TYPES::DOMAIN\_GLOBAL\_MAPPING\_-  
TYPE::LOCAL\_TYPE**

LOCAL\_TYPE(domain\_idx). The type of local for the domain\_idx'th domain for which the global number is mapped to a local number. The types depend on whether the mapped local number in the domain\_idx'th domain is an internal, boundary or ghost local number.

**See also:**

[DOMAIN\\_MAPPINGS::DomainType](#)

Definition at line 622 of file types.f90.

**7.192.2.4 INTEGER(INTG) TYPES::DOMAIN\_GLOBAL\_MAPPING\_TYPE::NUMBER\_OF\_-  
DOMAINS**

The number of domains that the global number is mapped to a local number in.

Definition at line 619 of file types.f90.

## 7.193 TYPES::DOMAIN\_LINE\_PTR\_TYPE Struct Reference

A buffer type to allow for an array of pointers to a [DOMAIN\\_LINE\\_TYPE](#).

Collaboration diagram for TYPES::DOMAIN\_LINE\_PTR\_TYPE:

### Public Attributes

- TYPE([DOMAIN\\_LINE\\_TYPE](#)), pointer PTR  
*A pointer to the domain line.*

#### 7.193.1 Detailed Description

A buffer type to allow for an array of pointers to a [DOMAIN\\_LINE\\_TYPE](#).

Definition at line 377 of file types.f90.

#### 7.193.2 Member Data Documentation

##### 7.193.2.1 TYPE(DOMAIN\_LINE\_TYPE),pointer TYPES::DOMAIN\_LINE\_PTR\_TYPE::PTR

A pointer to the domain line.

Definition at line 378 of file types.f90.

## 7.194 TYPES::DOMAIN\_LINE\_TYPE Struct Reference

Contains the information for a line in a domain.

Collaboration diagram for TYPES::DOMAIN\_LINE\_TYPE:

### Public Attributes

- INTEGER(INTG) **NUMBER**

*The line number in the domain.*

- TYPE(BASIS\_TYPE), pointer **BASIS**

*A pointer to the basis function for the line.*

- INTEGER(INTG), allocatable **NODES\_IN\_LINE**

*NODES\_IN\_LINE(nn). The local node number in the domain of the nn'th local node in the line. Old CMISS name NPL(2..5,nj,nl).*

- INTEGER(INTG), allocatable **DERIVATIVES\_IN\_LINE**

*DERIVATIVES\_IN\_LINE(nk,nn). The global derivative number of the local derivative nk for the local node nn in the line. Old CMISS name NPL(4..5,nj,nl).*

### 7.194.1 Detailed Description

Contains the information for a line in a domain.

Definition at line 369 of file types.f90.

### 7.194.2 Member Data Documentation

#### 7.194.2.1 TYPE(BASIS\_TYPE),pointer TYPES::DOMAIN\_LINE\_TYPE::BASIS

A pointer to the basis function for the line.

Definition at line 371 of file types.f90.

#### 7.194.2.2 INTEGER(INTG),allocatable TYPES::DOMAIN\_LINE\_TYPE::DERIVATIVES\_IN\_LINE

DERIVATIVES\_IN\_LINE(nk,nn). The global derivative number of the local derivative nk for the local node nn in the line. Old CMISS name NPL(4..5,nj,nl).

Definition at line 373 of file types.f90.

#### 7.194.2.3 INTEGER(INTG),allocatable TYPES::DOMAIN\_LINE\_TYPE::NODES\_IN\_LINE

NODES\_IN\_LINE(nn). The local node number in the domain of the nn'th local node in the line. Old CMISS name NPL(2..5,nj,nl).

Definition at line 372 of file types.f90.

**7.194.2.4 INTEGER(INTG) TYPES::DOMAIN\_LINE\_TYPE::NUMBER**

The line number in the domain.

Definition at line 370 of file types.f90.

## 7.195 TYPES::DOMAIN\_LINES\_TYPE Struct Reference

Contains the topology information for the lines of a domain.

Collaboration diagram for TYPES::DOMAIN\_LINES\_TYPE:

### Public Attributes

- TYPE(DOMAIN\_TYPE), pointer DOMAIN  
*The pointer to the domain for this lines topology information.*
- INTEGER(INTG) NUMBER\_OF\_LINES  
*The number of lines in this domain topology.*
- TYPE(DOMAIN\_LINE\_TYPE), allocatable LINES  
*LINES(nl). The pointer to the array of topology information for the lines of this domain. LINES(nl) contains the topological information for the nl'th local line of the domain.*

### 7.195.1 Detailed Description

Contains the topology information for the lines of a domain.

Definition at line 382 of file types.f90.

### 7.195.2 Member Data Documentation

#### 7.195.2.1 TYPE(DOMAIN\_TYPE),pointer TYPES::DOMAIN\_LINES\_TYPE::DOMAIN

The pointer to the domain for this lines topology information.

Definition at line 383 of file types.f90.

#### 7.195.2.2 TYPE(DOMAIN\_LINE\_TYPE),allocatable TYPES::DOMAIN\_LINES\_TYPE::LINES

LINES(nl). The pointer to the array of topology information for the lines of this domain. LINES(nl) contains the topological information for the nl'th local line of the domain.

Definition at line 385 of file types.f90.

#### 7.195.2.3 INTEGER(INTG) TYPES::DOMAIN\_LINES\_TYPE::NUMBER\_OF\_LINES

The number of lines in this domain topology.

Definition at line 384 of file types.f90.

## 7.196 TYPES::DOMAIN\_MAPPING\_TYPE Struct Reference

Contains information on the domain mappings (i.e., local and global numberings).

Collaboration diagram for TYPES::DOMAIN\_MAPPING\_TYPE:

### Public Attributes

- INTEGER(INTG) [NUMBER\\_OF\\_LOCAL](#)  
*The number of local numbers in the domain excluding ghost numbers.*
- INTEGER(INTG) [TOTAL\\_NUMBER\\_OF\\_LOCAL](#)  
*The total number of local numbers in the domain including ghost numbers.*
- INTEGER(INTG), allocatable [NUMBER\\_OF\\_DOMAIN\\_LOCAL](#)  
*NUMBER\_OF\_DOMAIN\_LOCAL(domain\_no). The total number of locals for domain\_no'th domain.  
 NOTE: the domain\_no goes from 0 to the number of domains-1.*
- INTEGER(INTG) [NUMBER\\_OF\\_GLOBAL](#)  
*The number of global numbers for this mapping.*
- INTEGER(INTG) [NUMBER\\_OF\\_DOMAINS](#)  
*The number of domains in this mapping.*
- INTEGER(INTG) [NUMBER\\_OF\\_INTERNAL](#)  
*The number of internal numbers in this mapping.*
- INTEGER(INTG), allocatable [INTERNAL\\_LIST](#)  
*INTERNAL\_LIST(i). The list of internal numbers for the mapping. The i'th position gives the i'th local internal number.*
- INTEGER(INTG) [NUMBER\\_OF\\_BOUNDARY](#)  
*The number of boundary numbers in this mapping.*
- INTEGER(INTG), allocatable [BOUNDARY\\_LIST](#)  
*BOUNDARY\_LIST(i). The list of boundary numbers for the mapping. The i'th position gives the i'th local boundary number.*
- INTEGER(INTG) [NUMBER\\_OF\\_GHOST](#)  
*The number of ghost numbers in this mapping.*
- INTEGER(INTG), allocatable [GHOST\\_LIST](#)  
*GHOST\_LIST(i). The list of ghost numbers for the mapping. The i'th position gives the i'th local ghost number.*
- INTEGER(INTG), allocatable [LOCAL\\_TO\\_GLOBAL\\_MAP](#)  
*LOCAL\_TO\_GLOBAL\_MAP(i). The global number for the i'th local number for the mapping.*
- TYPE([DOMAIN\\_GLOBAL\\_MAPPING\\_TYPE](#)), allocatable [GLOBAL\\_TO\\_LOCAL\\_MAP](#)  
*GLOBAL\_TO\_LOCAL\_MAP(i). The local information for the i'th global number for the mapping.*

- INTEGER(INTG) [NUMBER\\_OF\\_ADJACENT\\_DOMAINS](#)  
*The number of domains that are adjacent to this domain in the mapping.*
- INTEGER(INTG), allocatable [ADJACENT\\_DOMAINS\\_PTR](#)  
*ADJACENT\_DOMAINS\_PTR(domain\_no). The pointer to the list of adjacent domains for domain\_no. ADJACENT\_DOMAINS\_PTR(domain\_no) gives the starting position in ADJACENT\_DOMAINS\_LIST for the first adjacent domain number for domain number domain\_no. ADJACENT\_DOMAINS\_PTR(domain\_no+1) gives the last+1 position in ADJACENT\_DOMAINS\_LIST for the last adjacent domain number for domain number domain\_no. NOTE: the index for ADJACENT\_DOMAINS\_PTR varies from 0 to the number of domains+1.*
- INTEGER(INTG), allocatable [ADJACENT\\_DOMAINS\\_LIST](#)  
*ADJACENT\_DOMAINS\_LIST(i). The list of adjacent domains for each domain. The start and end positions for the list for domain number domain\_no are given by ADJACENT\_DOMAIN\_PTR(domain\_no) and ADJACENT\_DOMAIN\_PTR(domain\_no+1)-1 respectively.*
- TYPE([DOMAIN\\_ADJACENT\\_DOMAIN\\_TYPE](#)), allocatable [ADJACENT\\_DOMAINS](#)  
*ADJACENT\_DOMAINS(adjacent\_domain\_idx). The adjacent domain information for the adjacent\_domain\_idx'th adjacent domain to this domain.*

### 7.196.1 Detailed Description

Contains information on the domain mappings (i.e., local and global numberings).

Definition at line 626 of file types.f90.

### 7.196.2 Member Data Documentation

#### 7.196.2.1 TYPE([DOMAIN\\_ADJACENT\\_DOMAIN\\_TYPE](#)),allocatable [TYPES::DOMAIN\\_MAPPING\\_TYPE::ADJACENT\\_DOMAINS](#)

ADJACENT\_DOMAINS(adjacent\_domain\_idx). The adjacent domain information for the adjacent\_domain\_idx'th adjacent domain to this domain.

Definition at line 645 of file types.f90.

#### 7.196.2.2 INTEGER(INTG),allocatable [TYPES::DOMAIN\\_MAPPING\\_TYPE::ADJACENT\\_DOMAINS\\_LIST](#)

ADJACENT\_DOMAINS\_LIST(i). The list of adjacent domains for each domain. The start and end positions for the list for domain number domain\_no are given by ADJACENT\_DOMAIN\_PTR(domain\_no) and ADJACENT\_DOMAIN\_PTR(domain\_no+1)-1 respectively.

Definition at line 644 of file types.f90.

#### 7.196.2.3 INTEGER(INTG),allocatable [TYPES::DOMAIN\\_MAPPING\\_TYPE::ADJACENT\\_DOMAINS\\_PTR](#)

ADJACENT\_DOMAINS\_PTR(domain\_no). The pointer to the list of adjacent domains for domain\_no. ADJACENT\_DOMAINS\_PTR(domain\_no) gives the starting position in ADJACENT\_DOMAINS\_-

LIST for the first adjacent domain number for domain number domain\_no. ADJACENT\_DOMAINS\_PTR(domain\_no+1) gives the last+1 position in ADJACENT\_DOMAINS\_LIST for the last adjacent domain number for domain number domain\_no. NOTE: the index for ADJACENT\_DOMAINS\_PTR varies from 0 to the number of domains+1.

Definition at line 643 of file types.f90.

#### **7.196.2.4 INTEGER(INTG),allocatable TYPES::DOMAIN\_MAPPING\_TYPE::BOUNDARY\_LIST**

BOUNDARY\_LIST(i). The list of boundary numbers for the mapping. The i'th position gives the i'th local boundary number.

Definition at line 635 of file types.f90.

#### **7.196.2.5 INTEGER(INTG),allocatable TYPES::DOMAIN\_MAPPING\_TYPE::GHOST\_LIST**

GHOST\_LIST(i). The list of ghost numbers for the mapping. The i'th position gives the i'th local ghost number.

Definition at line 637 of file types.f90.

#### **7.196.2.6 TYPE(DOMAIN\_GLOBAL\_MAPPING\_TYPE),allocatable TYPES::DOMAIN\_MAPPING\_TYPE::GLOBAL\_TO\_LOCAL\_MAP**

GLOBAL\_TO\_LOCAL\_MAP(i). The local information for the i'th global number for the mapping.

Definition at line 641 of file types.f90.

#### **7.196.2.7 INTEGER(INTG),allocatable TYPES::DOMAIN\_MAPPING\_TYPE::INTERNAL\_LIST**

INTERNAL\_LIST(i). The list of internal numbers for the mapping. The i'th position gives the i'th local internal number.

Definition at line 633 of file types.f90.

#### **7.196.2.8 INTEGER(INTG),allocatable TYPES::DOMAIN\_MAPPING\_TYPE::LOCAL\_TO\_GLOBAL\_MAP**

LOCAL\_TO\_GLOBAL\_MAP(i). The global number for the i'th local number for the mapping.

Definition at line 640 of file types.f90.

#### **7.196.2.9 INTEGER(INTG) TYPES::DOMAIN\_MAPPING\_TYPE::NUMBER\_OF\_ADJACENT\_DOMAINS**

The number of domains that are adjacent to this domain in the mapping.

Definition at line 642 of file types.f90.

**7.196.2.10 INTEGER(INTG) TYPES::DOMAIN\_MAPPING\_TYPE::NUMBER\_OF\_BOUNDARY**

The number of boundary numbers in this mapping.

Definition at line 634 of file types.f90.

**7.196.2.11 INTEGER(INTG),allocatable TYPES::DOMAIN\_MAPPING\_TYPE::NUMBER\_OF\_DOMAIN\_LOCAL**

NUMBER\_OF\_DOMAIN\_LOCAL(domain\_no). The total number of locals for domain\_no'th domain.  
NOTE: the domain\_no goes from 0 to the number of domains-1.

Definition at line 629 of file types.f90.

**7.196.2.12 INTEGER(INTG) TYPES::DOMAIN\_MAPPING\_TYPE::NUMBER\_OF\_DOMAINS**

The number of domains in this mapping.

Definition at line 631 of file types.f90.

**7.196.2.13 INTEGER(INTG) TYPES::DOMAIN\_MAPPING\_TYPE::NUMBER\_OF\_GHOST**

The number of ghost numbers in this mapping.

Definition at line 636 of file types.f90.

**7.196.2.14 INTEGER(INTG) TYPES::DOMAIN\_MAPPING\_TYPE::NUMBER\_OF\_GLOBAL**

The number of global numbers for this mapping.

Definition at line 630 of file types.f90.

**7.196.2.15 INTEGER(INTG) TYPES::DOMAIN\_MAPPING\_TYPE::NUMBER\_OF\_INTERNAL**

The number of internal numbers in this mapping.

Definition at line 632 of file types.f90.

**7.196.2.16 INTEGER(INTG) TYPES::DOMAIN\_MAPPING\_TYPE::NUMBER\_OF\_LOCAL**

The number of local numbers in the domain excluding ghost numbers.

Definition at line 627 of file types.f90.

**7.196.2.17 INTEGER(INTG) TYPES::DOMAIN\_MAPPING\_TYPE::TOTAL\_NUMBER\_OF\_LOCAL**

The total number of local numbers in the domain including ghost numbers.

Definition at line 628 of file types.f90.

## 7.197 TYPES::DOMAIN\_MAPPINGS\_TYPE Struct Reference

Contains information on the domain decomposition mappings.

Collaboration diagram for TYPES::DOMAIN\_MAPPINGS\_TYPE:

### Public Attributes

- TYPE(DOMAIN\_TYPE), pointer DOMAIN  
*A pointer to the domain decomposition.*
- TYPE(DOMAIN\_MAPPING\_TYPE), pointer ELEMENTS  
*Pointer to the element mappings for the domain decomposition.*
- TYPE(DOMAIN\_MAPPING\_TYPE), pointer NODES  
*Pointer to the node mappings for the domain decomposition.*
- TYPE(DOMAIN\_MAPPING\_TYPE), pointer DOFS  
*Pointer to the dof mappings for the domain decomposition.*

### 7.197.1 Detailed Description

Contains information on the domain decomposition mappings.

Definition at line 649 of file types.f90.

### 7.197.2 Member Data Documentation

#### 7.197.2.1 TYPE(DOMAIN\_MAPPING\_TYPE),pointer TYPES::DOMAIN\_MAPPINGS\_- TYPE::DOFS

Pointer to the dof mappings for the domain decomposition.

Definition at line 653 of file types.f90.

#### 7.197.2.2 TYPE(DOMAIN\_TYPE),pointer TYPES::DOMAIN\_MAPPINGS\_TYPE::DOMAIN

A pointer to the domain decomposition.

Definition at line 650 of file types.f90.

#### 7.197.2.3 TYPE(DOMAIN\_MAPPING\_TYPE),pointer TYPES::DOMAIN\_MAPPINGS\_- TYPE::ELEMENTS

Pointer to the element mappings for the domain decomposition.

Definition at line 651 of file types.f90.

**7.197.2.4 TYPE(DOMAIN\_MAPPING\_TYPE),pointer TYPES::DOMAIN\_MAPPINGS\_-  
TYPE::NODES**

Pointer to the node mappings for the domain decomposition.

Definition at line 652 of file types.f90.

## 7.198 TYPES::DOMAIN\_NODE\_TYPE Struct Reference

Contains the topology information for a local node of a domain.

### Public Attributes

- INTEGER(INTG) [LOCAL\\_NUMBER](#)

*The local node number in the domain.*

- INTEGER(INTG) [GLOBAL\\_NUMBER](#)

*The corresponding global node number in the mesh of the local node number in the domain.*

- INTEGER(INTG) [USER\\_NUMBER](#)

*The corresponding user number for the node.*

- INTEGER(INTG) [NUMBER\\_OF\\_DERIVATIVES](#)

*The number of global derivatives at the node for the domain. Old [CMISS](#) name NKT(nj,np).*

- INTEGER(INTG), allocatable [PARTIAL\\_DERIVATIVE\\_INDEX](#)

*PARTIAL\_DERIVATIVE\_INDEX(nk). The partial derivative index (nu) of the nk'th global derivative for the node. Old [CMISS](#) name NUNK(nk,nj,np).*

- INTEGER(INTG), allocatable [DOF\\_INDEX](#)

*DOF\_INDEX(nk). The local dof derivative index (ny) in the domain of the nk'th global derivative for the node.*

- INTEGER(INTG) [NUMBER\\_OF\\_SURROUNDING\\_ELEMENTS](#)

*The number of elements surrounding the node in the domain. Old [CMISS](#) name NENP(np,0,0:nr).*

- INTEGER(INTG), pointer [SURROUNDING\\_ELEMENTS](#)

*SURROUNDING\_ELEMENTS(nep). The local element number of the nep'th element that is surrounding the node. Old [CMISS](#) name NENP(np,nep,0:nr).*

- INTEGER(INTG) [NUMBER\\_OF\\_NODE\\_LINES](#)

*The number of lines surrounding the node in the domain.*

- INTEGER(INTG), allocatable [NODE\\_LINES](#)

*NODE\_LINES(nlp). The local line number of the nlp'th line that is surrounding the node.*

### 7.198.1 Detailed Description

Contains the topology information for a local node of a domain.

Definition at line 428 of file types.f90.

## 7.198.2 Member Data Documentation

### 7.198.2.1 INTEGER(INTG),allocatable TYPES::DOMAIN\_NODE\_TYPE::DOF\_INDEX

DOF\_INDEX(nk). The local dof derivative index (ny) in the domain of the nk'th global derivative for the node.

Definition at line 434 of file types.f90.

### 7.198.2.2 INTEGER(INTG) TYPES::DOMAIN\_NODE\_TYPE::GLOBAL\_NUMBER

The corresponding global node number in the mesh of the local node number in the domain.

Definition at line 430 of file types.f90.

### 7.198.2.3 INTEGER(INTG) TYPES::DOMAIN\_NODE\_TYPE::LOCAL\_NUMBER

The local node number in the domain.

Definition at line 429 of file types.f90.

### 7.198.2.4 INTEGER(INTG),allocatable TYPES::DOMAIN\_NODE\_TYPE::NODE\_LINES

NODE\_LINES(nlp). The local line number of the nlp'th line that is surrounding the node.

Definition at line 438 of file types.f90.

### 7.198.2.5 INTEGER(INTG) TYPES::DOMAIN\_NODE\_TYPE::NUMBER\_OF\_DERIVATIVES

The number of global derivatives at the node for the domain. Old [CMISS](#) name NKT(nj,np).

Definition at line 432 of file types.f90.

### 7.198.2.6 INTEGER(INTG) TYPES::DOMAIN\_NODE\_TYPE::NUMBER\_OF\_NODE\_LINES

The number of lines surrounding the node in the domain.

Definition at line 437 of file types.f90.

### 7.198.2.7 INTEGER(INTG) TYPES::DOMAIN\_NODE\_TYPE::NUMBER\_OF\_- SURROUNDING\_ELEMENTS

The number of elements surrounding the node in the domain. Old [CMISS](#) name NENP(np,0,0:nr).

Definition at line 435 of file types.f90.

### 7.198.2.8 INTEGER(INTG),allocatable TYPES::DOMAIN\_NODE\_TYPE::PARTIAL\_- DERIVATIVE\_INDEX

PARTIAL\_DERIVATIVE\_INDEX(nk). The partial derivative index (nu) of the nk'th global derivative for the node. Old [CMISS](#) name NUNK(nk,nj,np).

Definition at line 433 of file types.f90.

**7.198.2.9 INTEGER(INTG),pointer TYPES::DOMAIN\_NODE\_TYPE::SURROUNDING\_ELEMENTS**

SURROUNDING\_ELEMENTS(nep). The local element number of the nep'th element that is surrounding the node. Old CMISS name NENP(np,nep,0:nr).

**Todo**

Change this to allocatable.

Definition at line 436 of file types.f90.

**7.198.2.10 INTEGER(INTG) TYPES::DOMAIN\_NODE\_TYPE::USER\_NUMBER**

The corresponding user number for the node.

Definition at line 431 of file types.f90.

## 7.199 TYPES::DOMAIN\_NODES\_TYPE Struct Reference

Contains the topology information for the nodes of a domain.

Collaboration diagram for TYPES::DOMAIN\_NODES\_TYPE:

### Public Attributes

- TYPE(DOMAIN\_TYPE), pointer DOMAIN  
*The pointer to the domain for this nodes topology information.*
- INTEGER(INTG) NUMBER\_OF\_NODES  
*The number of nodes (excluding ghost nodes) in this domain topology.*
- INTEGER(INTG) TOTAL\_NUMBER\_OF\_NODES  
*The total number of nodes (including ghost nodes) in this domain topology.*
- INTEGER(INTG) MAXIMUM\_NUMBER\_OF\_DERIVATIVES  
*The maximum number of derivatives over the nodes in this domain topology.*
- TYPE(DOMAIN\_NODE\_TYPE), pointer NODES  
*NODES(np). The pointer to the array of topology information for the nodes of this domain. NODES(np) contains the topological information for the np'th local node of the domain.*

### 7.199.1 Detailed Description

Contains the topology information for the nodes of a domain.

Definition at line 442 of file types.f90.

### 7.199.2 Member Data Documentation

#### 7.199.2.1 TYPE(DOMAIN\_TYPE),pointer TYPES::DOMAIN\_NODES\_TYPE::DOMAIN

The pointer to the domain for this nodes topology information.

Definition at line 443 of file types.f90.

#### 7.199.2.2 INTEGER(INTG) TYPES::DOMAIN\_NODES\_TYPE::MAXIMUM\_NUMBER\_OF\_- DERIVATIVES

The maximum number of derivatives over the nodes in this domain topology.

Definition at line 446 of file types.f90.

#### 7.199.2.3 TYPE(DOMAIN\_NODE\_TYPE),pointer TYPES::DOMAIN\_NODES\_TYPE::NODES

NODES(np). The pointer to the array of topology information for the nodes of this domain. NODES(np) contains the topological information for the np'th local node of the domain.

**Todo**

Change this to allocatable???

Definition at line 447 of file types.f90.

**7.199.2.4 INTEGER(INTG) TYPES::DOMAIN\_NODES\_TYPE::NUMBER\_OF\_NODES**

The number of nodes (excluding ghost nodes) in this domain topology.

Definition at line 444 of file types.f90.

**7.199.2.5 INTEGER(INTG) TYPES::DOMAIN\_NODES\_TYPE::TOTAL\_NUMBER\_OF\_NODES**

The total number of nodes (including ghost nodes) in this domain topology.

Definition at line 445 of file types.f90.

## 7.200 TYPES::DOMAIN\_PTR\_TYPE Struct Reference

A buffer type to allow for an array of pointers to a [DOMAIN\\_TYPE](#).

Collaboration diagram for TYPES::DOMAIN\_PTR\_TYPE:

### Public Attributes

- TYPE([DOMAIN\\_TYPE](#)), pointer PTR

*The pointer to the domain.*

### 7.200.1 Detailed Description

A buffer type to allow for an array of pointers to a [DOMAIN\\_TYPE](#).

Definition at line 669 of file types.f90.

### 7.200.2 Member Data Documentation

#### 7.200.2.1 TYPE(DOMAIN\_TYPE),pointer TYPES::DOMAIN\_PTR\_TYPE::PTR

The pointer to the domain.

Definition at line 670 of file types.f90.

## 7.201 TYPES::DOMAIN\_TOPOLOGY\_TYPE Struct Reference

Contains the topology information for a domain.

Collaboration diagram for TYPES::DOMAIN\_TOPOLOGY\_TYPE:

### Public Attributes

- TYPE(DOMAIN\_TYPE), pointer DOMAIN  
*The pointer to the domain for this topology information.*
- TYPE(DOMAIN\_NODES\_TYPE), pointer NODES  
*The pointer to the topology information for the nodes of this domain.*
- TYPE(DOMAIN\_DOFS\_TYPE), pointer DOFS  
*The pointer to the topology information for the dofs of this domain.*
- TYPE(DOMAIN\_ELEMENTS\_TYPE), pointer ELEMENTS  
*The pointer to the topology information for the elements of this domain.*
- TYPE(DOMAIN\_FACES\_TYPE), pointer FACES  
*The pointer to the topology information for the faces of this domain.*
- TYPE(DOMAIN\_LINES\_TYPE), pointer LINES  
*The pointer to the topology information for the lines of this domain.*

### 7.201.1 Detailed Description

Contains the topology information for a domain.

Definition at line 451 of file types.f90.

### 7.201.2 Member Data Documentation

#### 7.201.2.1 TYPE(DOMAIN\_DOFS\_TYPE),pointer TYPES::DOMAIN\_TOPOLOGY\_- TYPE::DOFS

The pointer to the topology information for the dofs of this domain.

Definition at line 454 of file types.f90.

#### 7.201.2.2 TYPE(DOMAIN\_TYPE),pointer TYPES::DOMAIN\_TOPOLOGY\_TYPE::DOMAIN

The pointer to the domain for this topology information.

Definition at line 452 of file types.f90.

**7.201.2.3 TYPE(DOMAIN\_ELEMENTS\_TYPE),pointer TYPES::DOMAIN\_TOPOLOGY\_-  
TYPE::ELEMENTS**

The pointer to the topology information for the elements of this domain.

Definition at line 455 of file types.f90.

**7.201.2.4 TYPE(DOMAIN\_FACES\_TYPE),pointer TYPES::DOMAIN\_TOPOLOGY\_-  
TYPE::FACES**

The pointer to the topology information for the faces of this domain.

Definition at line 456 of file types.f90.

**7.201.2.5 TYPE(DOMAIN\_LINES\_TYPE),pointer TYPES::DOMAIN\_TOPOLOGY\_-  
TYPE::LINES**

The pointer to the topology information for the lines of this domain.

Definition at line 457 of file types.f90.

**7.201.2.6 TYPE(DOMAIN\_NODES\_TYPE),pointer TYPES::DOMAIN\_TOPOLOGY\_-  
TYPE::NODES**

The pointer to the topology information for the nodes of this domain.

Definition at line 453 of file types.f90.

## 7.202 TYPES::DOMAIN\_TYPE Struct Reference

A pointer to the domain decomposition for this domain.

Collaboration diagram for TYPES::DOMAIN\_TYPE:

### Public Attributes

- TYPE(DECOMPOSITION\_TYPE), pointer DECOMPOSITION

*A pointer to the domain decomposition for this domain.*

- TYPE(MESH\_TYPE), pointer MESH

*A pointer to the mesh for this domain.*

- INTEGER(INTG) MESH\_COMPONENT\_NUMBER

*The mesh component number of the mesh which this domain was decomposed from.*

- TYPE(REGION\_TYPE), pointer REGION

*A pointer to the region that this domain is in. This is "inherited" from the mesh region.*

- INTEGER(INTG) NUMBER\_OF\_DIMENSIONS

*The number of dimensions for this domain. This is "inherited" from the mesh.*

- INTEGER(INTG), allocatable NODE\_DOMAIN

*NODE\_DOMAIN(np). The domain number that the np'th global node is in for the domain decomposition.*

*Note: the domain numbers start at 0 and go up to the NUMBER\_OF\_DOMAINS-1.*

- TYPE(DOMAIN\_MAPPINGS\_TYPE), pointer MAPPINGS

*Pointer to the mappings for the domain e.g., global to local and local to global maps.*

- TYPE(DOMAIN\_TOPOLOGY\_TYPE), pointer TOPOLOGY

*Pointer to the topology for the domain.*

### 7.202.1 Detailed Description

A pointer to the domain decomposition for this domain.

Definition at line 657 of file types.f90.

### 7.202.2 Member Data Documentation

#### 7.202.2.1 TYPE(DECOMPOSITION\_TYPE),pointer TYPES::DOMAIN\_TYPE::DECOMPOSITION

A pointer to the domain decomposition for this domain.

Definition at line 658 of file types.f90.

**7.202.2.2 TYPE(DOMAIN\_MAPPINGS\_TYPE),pointer TYPES::DOMAIN\_TYPE::MAPPINGS**

Pointer to the mappings for the domain e.g., global to local and local to global maps.

Definition at line 664 of file types.f90.

**7.202.2.3 TYPE(MESH\_TYPE),pointer TYPES::DOMAIN\_TYPE::MESH**

A pointer to the mesh for this domain.

Definition at line 659 of file types.f90.

**7.202.2.4 INTEGER(INTG) TYPES::DOMAIN\_TYPE::MESH\_COMPONENT\_NUMBER**

The mesh component number of the mesh which this domain was decomposed from.

Definition at line 660 of file types.f90.

**7.202.2.5 INTEGER(INTG),allocatable TYPES::DOMAIN\_TYPE::NODE\_DOMAIN**

NODE\_DOMAIN(np). The domain number that the np'th global node is in for the domain decomposition.  
Note: the domain numbers start at 0 and go up to the NUMBER\_OF\_DOMAINS-1.

Definition at line 663 of file types.f90.

**7.202.2.6 INTEGER(INTG) TYPES::DOMAIN\_TYPE::NUMBER\_OF\_DIMENSIONS**

The number of dimensions for this domain. This is "inherited" from the mesh.

Definition at line 662 of file types.f90.

**7.202.2.7 TYPE(REGION\_TYPE),pointer TYPES::DOMAIN\_TYPE::REGION**

A pointer to the region that this domain is in. This is "inherited" from the mesh region.

Definition at line 661 of file types.f90.

**7.202.2.8 TYPE(DOMAIN\_TOPOLOGY\_TYPE),pointer TYPES::DOMAIN\_TYPE::TOPOLOGY**

Pointer to the topology for the domain.

Definition at line 665 of file types.f90.

## 7.203 TYPES::EIGENPROBLEM\_SOLVER\_TYPE Struct Reference

Contains information for a time integration solver.

Collaboration diagram for TYPES::EIGENPROBLEM\_SOLVER\_TYPE:

### Public Attributes

- TYPE(SOLVER\_TYPE), pointer SOLVER  
*A pointer to the problem\_solver.*
- INTEGER(INTG) SOLVER\_LIBRARY  
*The library type for the eigenproblem solver.*

### 7.203.1 Detailed Description

Contains information for a time integration solver.

Definition at line 1445 of file types.f90.

### 7.203.2 Member Data Documentation

#### 7.203.2.1 TYPE(SOLVER\_TYPE),pointer TYPES::EIGENPROBLEM\_SOLVER\_TYPE::SOLVER

A pointer to the problem\_solver.

Definition at line 1446 of file types.f90.

#### 7.203.2.2 INTEGER(INTG) TYPES::EIGENPROBLEM\_SOLVER\_TYPE::SOLVER\_LIBRARY

The library type for the eigenproblem solver.

#### See also:

[SOLVER\\_ROUTINES::SolverLibraries](#), [SOLVER\\_ROUTINES](#)

Definition at line 1447 of file types.f90.

## 7.204 TYPES::ELEMENT\_MATRIX\_TYPE Struct Reference

Contains information for an element matrix.

### Public Attributes

- INTEGER(INTG) EQUATIONS\_MATRIX\_NUMBER
- INTEGER(INTG) NUMBER\_OF\_ROWS
- INTEGER(INTG) NUMBER\_OF\_COLUMNS
- INTEGER(INTG) MAX\_NUMBER\_OF\_ROWS
- INTEGER(INTG) MAX\_NUMBER\_OF\_COLUMNS
- INTEGER(INTG), allocatable ROW\_DOFS
- INTEGER(INTG), allocatable COLUMN\_DOFS
- REAL(DP), allocatable MATRIX

### 7.204.1 Detailed Description

Contains information for an element matrix.

Definition at line 952 of file types.f90.

### 7.204.2 Member Data Documentation

#### 7.204.2.1 INTEGER(INTG),allocatable TYPES::ELEMENT\_MATRIX\_TYPE::COLUMN\_DOFS

Definition at line 959 of file types.f90.

#### 7.204.2.2 INTEGER(INTG) TYPES::ELEMENT\_MATRIX\_TYPE::EQUATIONS\_MATRIX\_NUMBER

Definition at line 953 of file types.f90.

#### 7.204.2.3 REAL(DP),allocatable TYPES::ELEMENT\_MATRIX\_TYPE::MATRIX

Definition at line 960 of file types.f90.

#### 7.204.2.4 INTEGER(INTG) TYPES::ELEMENT\_MATRIX\_TYPE::MAX\_NUMBER\_OF\_COLUMNS

Definition at line 957 of file types.f90.

#### 7.204.2.5 INTEGER(INTG) TYPES::ELEMENT\_MATRIX\_TYPE::MAX\_NUMBER\_OF\_ROWS

Definition at line 956 of file types.f90.

**7.204.2.6 INTEGER(INTG) TYPES::ELEMENT\_MATRIX\_TYPE::NUMBER\_OF\_COLUMNS**

Definition at line 955 of file types.f90.

**7.204.2.7 INTEGER(INTG) TYPES::ELEMENT\_MATRIX\_TYPE::NUMBER\_OF\_ROWS**

Definition at line 954 of file types.f90.

**7.204.2.8 INTEGER(INTG),allocatable TYPES::ELEMENT\_MATRIX\_TYPE::ROW\_DOFS**

Definition at line 958 of file types.f90.

## 7.205 TYPES::ELEMENT\_VECTOR\_TYPE Struct Reference

Contains information for an element vector.

### Public Attributes

- INTEGER(INTG) [NUMBER\\_OF\\_ROWS](#)
- INTEGER(INTG) [MAX\\_NUMBER\\_OF\\_ROWS](#)
- INTEGER(INTG), allocatable [ROW\\_DOFS](#)
- REAL(DP), allocatable [VECTOR](#)

### 7.205.1 Detailed Description

Contains information for an element vector.

Definition at line 964 of file types.f90.

### 7.205.2 Member Data Documentation

#### 7.205.2.1 INTEGER(INTG) TYPES::ELEMENT\_VECTOR\_TYPE::MAX\_NUMBER\_OF\_ROWS

Definition at line 966 of file types.f90.

#### 7.205.2.2 INTEGER(INTG) TYPES::ELEMENT\_VECTOR\_TYPE::NUMBER\_OF\_ROWS

Definition at line 965 of file types.f90.

#### 7.205.2.3 INTEGER(INTG),allocatable TYPES::ELEMENT\_VECTOR\_TYPE::ROW\_DOFS

Definition at line 967 of file types.f90.

#### 7.205.2.4 REAL(DP),allocatable TYPES::ELEMENT\_VECTOR\_TYPE::VECTOR

Definition at line 968 of file types.f90.

## 7.206 TYPES::EQUATIONS\_COL\_TO\_SOLVER\_COLS\_MAP\_- TYPE Struct Reference

### Public Attributes

- INTEGER(INTG) [NUMBER\\_OF\\_SOLVER\\_COLS](#)  
*The number of solver cols this equations column is mapped to.*
- INTEGER(INTG), allocatable [SOLVER\\_COLS](#)  
*SOLVER\_COLS(i). The solver column number for the i'th solver column that this column is mapped to.*
- REAL(DP), allocatable [COUPLING\\_COEFFICIENTS](#)  
*COUPLING\_COEFFICIENTS(i). The coupling coefficients for the i'th solver column that this column is mapped to.*

### 7.206.1 Detailed Description

Definition at line 1471 of file types.f90.

### 7.206.2 Member Data Documentation

#### 7.206.2.1 REAL(DP),allocatable TYPES::EQUATIONS\_COL\_TO\_SOLVER\_COLS\_MAP\_- TYPE::COUPLING\_COEFFICIENTS

COUPLING\_COEFFICIENTS(i). The coupling coefficients for the i'th solver column that this column is mapped to.

Definition at line 1474 of file types.f90.

#### 7.206.2.2 INTEGER(INTG) TYPES::EQUATIONS\_COL\_TO\_SOLVER\_COLS\_MAP\_- TYPE::NUMBER\_OF\_SOLVER\_COLS

The number of solver cols this equations column is mapped to.

Definition at line 1472 of file types.f90.

#### 7.206.2.3 INTEGER(INTG),allocatable TYPES::EQUATIONS\_COL\_TO\_SOLVER\_COLS\_- MAP\_TYPE::SOLVER\_COLS

SOLVER\_COLS(i). The solver column number for the i'th solver column that this column is mapped to.

Definition at line 1473 of file types.f90.

## 7.207 TYPES::EQUATIONS\_INTERPOLATION\_TYPE Struct Reference

Contains information on the interpolation for the equations.

Collaboration diagram for TYPES::EQUATIONS\_INTERPOLATION\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_TYPE), pointer EQUATIONS  
*A pointer to the equations.*
- TYPE(FIELD\_TYPE), pointer GEOMETRIC\_FIELD  
*A pointer to the geometric field for the equations.*
- TYPE(FIELD\_TYPE), pointer FIBRE\_FIELD  
*A pointer to the fibre field for the equations (if one is defined).*
- TYPE(FIELD\_TYPE), pointer DEPENDENT\_FIELD  
*A pointer to the dependent field for the equations.*
- TYPE(FIELD\_TYPE), pointer MATERIAL\_FIELD  
*A pointer to the material field for the equations (if one is defined).*
- TYPE(FIELD\_TYPE), pointer SOURCE\_FIELD  
*A pointer to the source field for the equations (if one is defined).*
- TYPE(FIELD\_INTERPOLATION\_PARAMETERS\_TYPE), pointer GEOMETRIC\_INTERP\_PARAMETERS  
*A pointer to the geometric interpolation parameters for the equations.*
- TYPE(FIELD\_INTERPOLATION\_PARAMETERS\_TYPE), pointer FIBRE\_INTERP\_PARAMETERS  
*A pointer to the fibre interpolation parameters for the equations (if a fibre field is defined).*
- TYPE(FIELD\_INTERPOLATION\_PARAMETERS\_TYPE), pointer DEPENDENT\_INTERP\_PARAMETERS  
*A pointer to the dependent interpolation parameters for the equations.*
- TYPE(FIELD\_INTERPOLATION\_PARAMETERS\_TYPE), pointer MATERIAL\_INTERP\_PARAMETERS  
*A pointer to the material interpolation parameters for the equations (if a material field is defined).*
- TYPE(FIELD\_INTERPOLATION\_PARAMETERS\_TYPE), pointer SOURCE\_INTERP\_PARAMETERS  
*A pointer to the source interpolation parameters for the equations (if a source field is defined).*
- TYPE(FIELD\_INTERPOLATED\_POINT\_TYPE), pointer GEOMETRIC\_INTERP\_POINT  
*A pointer to the geometric interpolated point information for the equations.*

- TYPE(FIELD\_INTERPOLATED\_POINT\_TYPE), pointer FIBRE\_INTERP\_POINT  
*A pointer to the fibre interpolated point information for the equations (if a fibre field is defined).*
- TYPE(FIELD\_INTERPOLATED\_POINT\_TYPE), pointer DEPENDENT\_INTERP\_POINT  
*A pointer to the dependent interpolated point information for the equations.*
- TYPE(FIELD\_INTERPOLATED\_POINT\_TYPE), pointer MATERIAL\_INTERP\_POINT  
*A pointer to the material interpolated point information for the equations (if a material field is defined).*
- TYPE(FIELD\_INTERPOLATED\_POINT\_TYPE), pointer SOURCE\_INTERP\_POINT  
*A pointer to the source interpolated point information for the equations (if a source field is defined).*
- TYPE(FIELD\_INTERPOLATED\_POINT\_METRICS\_TYPE), pointer GEOMETRIC\_INTERP\_POINT\_METRICS  
*A pointer to the geometric interpolated point metrics information.*
- TYPE(FIELD\_INTERPOLATED\_POINT\_METRICS\_TYPE), pointer FIBRE\_INTERP\_POINT\_METRICS  
*A pointer to the fibre interpolated point metrics information.*

### 7.207.1 Detailed Description

Contains information on the interpolation for the equations.

Definition at line 1178 of file types.f90.

### 7.207.2 Member Data Documentation

#### 7.207.2.1 TYPE(FIELD\_TYPE),pointer TYPES::EQUATIONS\_INTERPOLATION\_TYPE::DEPENDENT\_FIELD

A pointer to the dependent field for the equations.

Definition at line 1182 of file types.f90.

#### 7.207.2.2 TYPE(FIELD\_INTERPOLATION\_PARAMETERS\_TYPE),pointer TYPES::EQUATIONS\_INTERPOLATION\_TYPE::DEPENDENT\_INTERP\_PARAMETERS

A pointer to the dependent interpolation parameters for the equations.

Definition at line 1187 of file types.f90.

#### 7.207.2.3 TYPE(FIELD\_INTERPOLATED\_POINT\_TYPE),pointer TYPES::EQUATIONS\_INTERPOLATION\_TYPE::DEPENDENT\_INTERP\_POINT

A pointer to the dependent interpolated point information for the equations.

Definition at line 1192 of file types.f90.

**7.207.2.4 TYPE(EQUATIONS\_TYPE),pointer TYPES::EQUATIONS\_INTERPOLATION\_-  
TYPE::EQUATIONS**

A pointer to the equations.

Definition at line 1179 of file types.f90.

**7.207.2.5 TYPE(FIELD\_TYPE),pointer TYPES::EQUATIONS\_INTERPOLATION\_-  
TYPE::FIBRE\_FIELD**

A pointer to the fibre field for the equations (if one is defined).

Definition at line 1181 of file types.f90.

**7.207.2.6 TYPE(FIELD\_INTERPOLATION\_PARAMETERS\_TYPE),pointer  
TYPES::EQUATIONS\_INTERPOLATION\_TYPE::FIBRE\_INTERP\_PARAMETERS**

A pointer to the fibre interpolation parameters for the equations (if a fibre field is defined).

Definition at line 1186 of file types.f90.

**7.207.2.7 TYPE(FIELD\_INTERPOLATED\_POINT\_TYPE),pointer  
TYPES::EQUATIONS\_INTERPOLATION\_TYPE::FIBRE\_INTERP\_POINT**

A pointer to the fibre interpolated point information for the equations (if a fibre field is defined).

Definition at line 1191 of file types.f90.

**7.207.2.8 TYPE(FIELD\_INTERPOLATED\_POINT\_METRICS\_TYPE),pointer  
TYPES::EQUATIONS\_INTERPOLATION\_TYPE::FIBRE\_INTERP\_POINT\_-  
METRICS**

A pointer to the fibre interpolated point metrics information.

Definition at line 1196 of file types.f90.

**7.207.2.9 TYPE(FIELD\_TYPE),pointer TYPES::EQUATIONS\_INTERPOLATION\_-  
TYPE::GEOMETRIC\_FIELD**

A pointer to the geometric field for the equations.

Definition at line 1180 of file types.f90.

**7.207.2.10 TYPE(FIELD\_INTERPOLATION\_PARAMETERS\_TYPE),pointer  
TYPES::EQUATIONS\_INTERPOLATION\_TYPE::GEOMETRIC\_INTERP\_-  
PARAMETERS**

A pointer to the geometric interpolation parameters for the equations.

Definition at line 1185 of file types.f90.

**7.207.2.11 TYPE(FIELD\_INTERPOLATED\_POINT\_TYPE),pointer  
TYPES::EQUATIONS\_INTERPOLATION\_TYPE::GEOMETRIC\_INTERP\_POINT**

A pointer to the geometric interpolated point information for the equations.

Definition at line 1190 of file types.f90.

**7.207.2.12 TYPE(FIELD\_INTERPOLATED\_POINT\_METRICS\_TYPE),pointer  
TYPES::EQUATIONS\_INTERPOLATION\_TYPE::GEOMETRIC\_INTERP\_-  
POINT\_METRICS**

A pointer to the geometric interpolated point metrics information.

Definition at line 1195 of file types.f90.

**7.207.2.13 TYPE(FIELD\_TYPE),pointer TYPES::EQUATIONS\_INTERPOLATION\_-  
TYPE::MATERIAL\_FIELD**

A pointer to the material field for the equations (if one is defined).

Definition at line 1183 of file types.f90.

**7.207.2.14 TYPE(FIELD\_INTERPOLATION\_PARAMETERS\_TYPE),pointer  
TYPES::EQUATIONS\_INTERPOLATION\_TYPE::MATERIAL\_INTERP\_-  
PARAMETERS**

A pointer to the material interpolation parameters for the equations (if a material field is defined).

Definition at line 1188 of file types.f90.

**7.207.2.15 TYPE(FIELD\_INTERPOLATED\_POINT\_TYPE),pointer  
TYPES::EQUATIONS\_INTERPOLATION\_TYPE::MATERIAL\_INTERP\_POINT**

A pointer to the material interpolated point information for the equations (if a material field is defined).

Definition at line 1193 of file types.f90.

**7.207.2.16 TYPE(FIELD\_TYPE),pointer TYPES::EQUATIONS\_INTERPOLATION\_-  
TYPE::SOURCE\_FIELD**

A pointer to the source field for the equations (if one is defined).

Definition at line 1184 of file types.f90.

**7.207.2.17 TYPE(FIELD\_INTERPOLATION\_PARAMETERS\_TYPE),pointer  
TYPES::EQUATIONS\_INTERPOLATION\_TYPE::SOURCE\_INTERP\_-  
PARAMETERS**

A pointer to the source interpolation parameters for the equations (if a source field is defined).

Definition at line 1189 of file types.f90.

**7.207.2.18 TYPE(FIELD\_INTERPOLATED\_POINT\_TYPE),pointer  
TYPES::EQUATIONS\_INTERPOLATION\_TYPE::SOURCE\_INTERP\_POINT**

A pointer to the source interpolated point information for the equations (if a source field is defined).

Definition at line 1194 of file types.f90.

## 7.208 TYPES::EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP\_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP\_TYPE:

### Public Attributes

- INTEGER(INTG) [VARIABLE\\_TYPE](#)  
*The dependent variable type mapped to this equations matrix.*
- TYPE([FIELD\\_VARIABLE\\_TYPE](#)), pointer [VARIABLE](#)  
*A pointer to the field variable that is mapped to this equations matrix.*
- TYPE([EQUATIONS\\_JACOBIAN\\_TYPE](#)), pointer [JACOBIAN](#)  
*A pointer to the equations matrix.*
- INTEGER(INTG) [NUMBER\\_OF\\_COLUMNS](#)  
*The number of columns in this equations matrix.*
- REAL(DP) [JACOBIAN\\_COEFFICIENT](#)  
*The multiplicative coefficient for the matrix in the equation set.*
- INTEGER(INTG), allocatable [EQUATIONS\\_COLUMN\\_TO\\_DOF\\_VARIABLE\\_MAP](#)  
*COLUMN\_TO\_DOF\_MAP(column\_idx). The variable DOF that the column\_idx'th column of this equations matrix is mapped to.*
- TYPE([DOMAIN\\_MAPPING\\_TYPE](#)), pointer [COLUMN\\_DOFS\\_MAPPING](#)  
*A pointer to the column dofs domain mapping for the matrix variable.*

### 7.208.1 Detailed Description

Definition at line 1088 of file types.f90.

### 7.208.2 Member Data Documentation

#### 7.208.2.1 TYPE(DOMAIN\_MAPPING\_TYPE),pointer TYPES::EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP\_TYPE::COLUMN\_DOFS\_MAPPING

A pointer to the column dofs domain mapping for the matrix variable.

Definition at line 1095 of file types.f90.

#### 7.208.2.2 INTEGER(INTG),allocatable TYPES::EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP\_TYPE::EQUATIONS\_COLUMN\_TO\_DOF\_VARIABLE\_MAP

COLUMN\_TO\_DOF\_MAP(column\_idx). The variable DOF that the column\_idx'th column of this equations matrix is mapped to.

Definition at line 1094 of file types.f90.

**7.208.2.3 TYPE(EQUATIONS\_JACOBIAN\_TYPE),pointer TYPES::EQUATIONS\_-  
JACOBIAN\_TO\_VARIABLE\_MAP\_TYPE::JACOBIAN**

A pointer to the equations matrix.

Definition at line 1091 of file types.f90.

**7.208.2.4 REAL(DP) TYPES::EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP\_-  
TYPE::JACOBIAN\_COEFFICIENT**

The multiplicative coefficient for the matrix in the equation set.

Definition at line 1093 of file types.f90.

**7.208.2.5 INTEGER(INTG) TYPES::EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP\_-  
TYPE::NUMBER\_OF\_COLUMNS**

The number of columns in this equations matrix.

Definition at line 1092 of file types.f90.

**7.208.2.6 TYPE(FIELD\_VARIABLE\_TYPE),pointer TYPES::EQUATIONS\_JACOBIAN\_TO\_-  
VARIABLE\_MAP\_TYPE::VARIABLE**

A pointer to the field variable that is mapped to this equations matrix.

Definition at line 1090 of file types.f90.

**7.208.2.7 INTEGER(INTG) TYPES::EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP\_-  
TYPE::VARIABLE\_TYPE**

The dependent variable type mapped to this equations matrix.

Definition at line 1089 of file types.f90.

## 7.209 TYPES::EQUATIONS\_JACOBIAN\_TYPE Struct Reference

Contains information on the Jacobian matrix for nonlinear problems.

Collaboration diagram for TYPES::EQUATIONS\_JACOBIAN\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_MATRICES\_NONLINEAR\_TYPE), pointer NONLINEAR\_MATRICES
- INTEGER(INTG) STORAGE\_TYPE

*The storage (sparsity) type for this matrix.*

- INTEGER(INTG) STRUCTURE\_TYPE

*The structure (sparsity) type for this matrix.*

- INTEGER(INTG) NUMBER\_OF\_COLUMNS

*The number of columns in this global matrix.*

- LOGICAL JACOBIAN\_CALCULATION\_TYPE

*The type of how the Jacobian is calculated.*

- LOGICAL UPDATE\_JACOBIAN

*Is .TRUE. if this Jacobian matrix is to be updated.*

- TYPE(DISTRIBUTED\_MATRIX\_TYPE), pointer JACOBIAN

*A pointer to the distributed jacobian matrix data.*

- TYPE(ELEMENT\_MATRIX\_TYPE) ELEMENT\_JACOBIAN

*The element matrix for this Jacobian matrix. This is not used if the Jacobian is not supplied.*

- TYPE(ELEMENT\_VECTOR\_TYPE) ELEMENT\_RESIDUAL

*An element residual vector for nonlinear problems when Jacobians are calculated. This is not used if the Jacobian is not supplied. Old CMISS name RE2.*

### 7.209.1 Detailed Description

Contains information on the Jacobian matrix for nonlinear problems.

Definition at line 988 of file types.f90.

### 7.209.2 Member Data Documentation

#### 7.209.2.1 TYPE(ELEMENT\_MATRIX\_TYPE) TYPES::EQUATIONS\_JACOBIAN\_- TYPE::ELEMENT\_JACOBIAN

The element matrix for this Jacobian matrix. This is not used if the Jacobian is not supplied.

Definition at line 996 of file types.f90.

**7.209.2.2 TYPE(ELEMENT\_VECTOR\_TYPE) TYPES::EQUATIONS\_JACOBIAN\_-  
TYPE::ELEMENT\_RESIDUAL**

An element residual vector for nonlinear problems when Jacobians are calculated. This is not used if the Jacobian is not supplied. Old [CMISS](#) name RE2.

Definition at line 997 of file types.f90.

**7.209.2.3 TYPE(DISTRIBUTED\_MATRIX\_TYPE),pointer TYPES::EQUATIONS\_-  
JACOBIAN\_TYPE::JACOBIAN**

A pointer to the distributed jacobian matrix data.

Definition at line 995 of file types.f90.

**7.209.2.4 LOGICAL TYPES::EQUATIONS\_JACOBIAN\_TYPE::JACOBIAN\_-  
CALCULATION\_TYPE**

The type of how the Jacobian is calculated.

**See also:**

[EQUATIONS\\_SET\\_CONSTANTS::JacobianCalculationTypes](#),[EQUATIONS\\_SET\\_CONSTANTS](#)

Definition at line 993 of file types.f90.

**7.209.2.5 TYPE(EQUATIONS\_MATRICES\_NONLINEAR\_TYPE),pointer  
TYPES::EQUATIONS\_JACOBIAN\_TYPE::NONLINEAR\_MATRICES**

Definition at line 989 of file types.f90.

**7.209.2.6 INTEGER(INTG) TYPES::EQUATIONS\_JACOBIAN\_TYPE::NUMBER\_OF\_-  
COLUMNS**

The number of columns in this global matrix.

Definition at line 992 of file types.f90.

**7.209.2.7 INTEGER(INTG) TYPES::EQUATIONS\_JACOBIAN\_TYPE::STORAGE\_TYPE**

The storage (sparsity) type for this matrix.

Definition at line 990 of file types.f90.

**7.209.2.8 INTEGER(INTG) TYPES::EQUATIONS\_JACOBIAN\_TYPE::STRUCTURE\_TYPE**

The structure (sparsity) type for this matrix.

Definition at line 991 of file types.f90.

**7.209.2.9 LOGICAL TYPES::EQUATIONS\_JACOBIAN\_TYPE::UPDATE\_JACOBIAN**

Is .TRUE. if this Jacobian matrix is to be updated.

Definition at line 994 of file types.f90.

## 7.210 TYPES::EQUATIONS\_LINEAR\_DATA\_TYPE Struct Reference

Contains information on any data required for a linear solution.

Collaboration diagram for TYPES::EQUATIONS\_LINEAR\_DATA\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_TYPE), pointer EQUATIONS

*A pointer to the equations.*

#### 7.210.1 Detailed Description

Contains information on any data required for a linear solution.

Definition at line 1200 of file types.f90.

#### 7.210.2 Member Data Documentation

##### 7.210.2.1 TYPE(EQUATIONS\_TYPE),pointer TYPES::EQUATIONS\_LINEAR\_DATA\_TYPE::EQUATIONS

A pointer to the equations.

Definition at line 1201 of file types.f90.

## 7.211 TYPES::EQUATIONS\_MAPPING\_CREATE\_VALUES\_- CACHE\_TYPE Struct Reference

### Public Attributes

- INTEGER(INTG) [NUMBER\\_OF\\_LINEAR\\_EQUATIONS\\_MATRICES](#)

*The number of linear matrices in the equations mapping.*
- INTEGER(INTG), allocatable [MATRIX\\_VARIABLE\\_TYPES](#)

*MATRIX\_VARIABLE\_TYPES(matrix\_idx). The dependent variable type mapped to the matrix\_idx'th equations matrix.*
- REAL(DP), allocatable [MATRIX\\_COEFFICIENTS](#)

*MATRIX\_COEFFICIENTS(matrix\_idx). The coefficient of the matrix\_idx'th matrix in the equations set.*
- INTEGER(INTG) [RESIDUAL\\_VARIABLE\\_TYPE](#)

*The dependent variable type mapped to the residual vector.*
- REAL(DP) [RESIDUAL\\_COEFFICIENT](#)

*The coefficient multiplying the residual vector.*
- INTEGER(INTG) [RHS\\_VARIABLE\\_TYPE](#)

*The dependent variable type mapped to the rhs vector.*
- REAL(DP) [RHS\\_COEFFICIENT](#)

*The coefficient multiplying the RHS vector.*
- INTEGER(INTG) [SOURCE\\_VARIABLE\\_TYPE](#)

*The source variable type mapped to the source vector.*
- REAL(DP) [SOURCE\\_COEFFICIENT](#)

*The coefficient multiplying the source vector.*

### 7.211.1 Detailed Description

Definition at line 1138 of file types.f90.

### 7.211.2 Member Data Documentation

#### 7.211.2.1 REAL(DP),allocatable TYPES::EQUATIONS\_MAPPING\_CREATE\_VALUES\_- CACHE\_TYPE::MATRIX\_COEFFICIENTS

[MATRIX\\_COEFFICIENTS\(matrix\\_idx\)](#). The coefficient of the matrix\_idx'th matrix in the equations set.

Definition at line 1141 of file types.f90.

**7.211.2.2 INTEGER(INTG),allocatable TYPES::EQUATIONS\_MAPPING\_CREATE\_-  
VALUES\_CACHE\_TYPE::MATRIX\_VARIABLE\_TYPES**

MATRIX\_VARIABLE\_TYPES(matrix\_idx). The dependent variable type mapped to the matrix\_idx'th equations matrix.

Definition at line 1140 of file types.f90.

**7.211.2.3 INTEGER(INTG) TYPES::EQUATIONS\_MAPPING\_CREATE\_VALUES\_CACHE\_-  
TYPE::NUMBER\_OF\_LINEAR\_EQUATIONS\_MATRICES**

The number of linear matrices in the equations mapping.

Definition at line 1139 of file types.f90.

**7.211.2.4 REAL(DP) TYPES::EQUATIONS\_MAPPING\_CREATE\_VALUES\_CACHE\_-  
TYPE::RESIDUAL\_COEFFICIENT**

The coefficient multiplying the residual vector.

Definition at line 1143 of file types.f90.

**7.211.2.5 INTEGER(INTG) TYPES::EQUATIONS\_MAPPING\_CREATE\_VALUES\_CACHE\_-  
TYPE::RESIDUAL\_VARIABLE\_TYPE**

The dependent variable type mapped to the residual vector.

Definition at line 1142 of file types.f90.

**7.211.2.6 REAL(DP) TYPES::EQUATIONS\_MAPPING\_CREATE\_VALUES\_CACHE\_-  
TYPE::RHS\_COEFFICIENT**

The coefficient multiplying the RHS vector.

Definition at line 1145 of file types.f90.

**7.211.2.7 INTEGER(INTG) TYPES::EQUATIONS\_MAPPING\_CREATE\_VALUES\_CACHE\_-  
TYPE::RHS\_VARIABLE\_TYPE**

The dependent variable type mapped to the rhs vector.

Definition at line 1144 of file types.f90.

**7.211.2.8 REAL(DP) TYPES::EQUATIONS\_MAPPING\_CREATE\_VALUES\_CACHE\_-  
TYPE::SOURCE\_COEFFICIENT**

The coefficient multiplying the source vector.

Definition at line 1147 of file types.f90.

## **7.211 TYPES::EQUATIONS\_MAPPING\_CREATE\_VALUES\_CACHE\_TYPE Struct Reference**

### **7.211.2.9 INTEGER(INTG) TYPES::EQUATIONS\_MAPPING\_CREATE\_VALUES\_CACHE\_- TYPE::SOURCE\_VARIABLE\_TYPE**

The source variable type mapped to the source vector.

Definition at line 1146 of file types.f90.

## 7.212 TYPES::EQUATIONS\_MAPPING\_LINEAR\_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS\_MAPPING\_LINEAR\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_MAPPING\_TYPE), pointer EQUATIONS\_MAPPING  
*A pointer to the equations mapping.*
- INTEGER(INTG) NUMBER\_OF\_LINEAR\_EQUATIONS\_MATRICES  
*The number of linear equations matrices in this mapping.*
- INTEGER(INTG) NUMBER\_OF\_LINEAR\_MATRIX\_VARIABLES  
*The number of dependent variables involved in the linear equations matrix mapping.*
- INTEGER(INTG), allocatable MATRIX\_VARIABLE\_TYPES  
*MATRIX\_VARIABLE\_TYPES(i). The variable type of the i'th variable type involved in the equations matrix mapping.*
- TYPE(VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP\_TYPE), allocatable VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAPS  
*VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAPS(variable\_type\_idx). The equations matrices mapping for the variable\_type\_idx'th variable type.*
- TYPE(EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_TYPE), allocatable EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAPS  
*EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAPS(matrix\_idx). The mappings for the matrix\_idx'th equations matrix.*
- INTEGER(INTG), allocatable EQUATIONS\_ROW\_TO\_VARIABLE\_DOF\_MAPS  
*EQUATIONS\_ROW\_TO\_VARIABLE\_DOF\_MAPS(row\_idx,variable\_type\_idx). The row mappings for the row\_idx'th row of the equations matrices to the variable\_type\_idx'th variable.*

### 7.212.1 Detailed Description

Definition at line 1077 of file types.f90.

### 7.212.2 Member Data Documentation

#### 7.212.2.1 TYPE(EQUATIONS\_MAPPING\_TYPE),pointer TYPES::EQUATIONS\_MAPPING\_LINEAR\_TYPE::EQUATIONS\_MAPPING

A pointer to the equations mapping.

Definition at line 1078 of file types.f90.

**7.212.2.2 TYPE(EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_TYPE),allocatable  
TYPES::EQUATIONS\_MAPPING\_LINEAR\_TYPE::EQUATIONS\_MATRIX\_TO\_-  
VARIABLE\_MAPS**

EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAPS(matrix\_idx). The mappings for the matrix\_idx'th equations matrix.

Definition at line 1084 of file types.f90.

**7.212.2.3 INTEGER(INTG),allocatable TYPES::EQUATIONS\_MAPPING\_LINEAR\_-  
TYPE::EQUATIONS\_ROW\_TO\_VARIABLE\_DOF\_MAPS**

EQUATIONS\_ROW\_TO\_VARIABLE\_DOF\_MAPS(row\_idx,variable\_type\_idx). The row mappings for the row\_idx'th row of the equations matrices to the variable\_type\_idx'th variable.

Definition at line 1085 of file types.f90.

**7.212.2.4 INTEGER(INTG),allocatable TYPES::EQUATIONS\_MAPPING\_LINEAR\_-  
TYPE::MATRIX\_VARIABLE\_TYPES**

MATRIX\_VARIABLE\_TYPES(i). The variable type of the i'th variable type involved in the equations matrix mapping.

Definition at line 1081 of file types.f90.

**7.212.2.5 INTEGER(INTG) TYPES::EQUATIONS\_MAPPING\_LINEAR\_TYPE::NUMBER\_-  
OF\_LINEAR\_EQUATIONS\_MATRICES**

The number of linear equations matrices in this mapping.

Definition at line 1079 of file types.f90.

**7.212.2.6 INTEGER(INTG) TYPES::EQUATIONS\_MAPPING\_LINEAR\_TYPE::NUMBER\_-  
OF\_LINEAR\_MATRIX\_VARIABLES**

The number of dependent variables involved in the linear equations matrix mapping.

Definition at line 1080 of file types.f90.

**7.212.2.7 TYPE(VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP\_TYPE),allocatable  
TYPES::EQUATIONS\_MAPPING\_LINEAR\_TYPE::VARIABLE\_TO\_EQUATIONS\_-  
MATRICES\_MAPS**

VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAPS(variable\_type\_idx). The equations matrices mapping for the variable\_type\_idx'th variable type.

Definition at line 1083 of file types.f90.

## 7.213 TYPES::EQUATIONS\_MAPPING\_NONLINEAR\_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS\_MAPPING\_NONLINEAR\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_MAPPING\_TYPE), pointer EQUATIONS\_MAPPING  
*A pointer to the equations mapping.*
- TYPE(VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_MAP\_TYPE) VARIABLE\_TO\_-  
 JACOBIAN\_MAP
- TYPE(EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP\_TYPE) JACOBIAN\_TO\_-  
 VARIABLE\_MAP
- INTEGER(INTG) RESIDUAL\_VARIABLE\_TYPE
- TYPE(FIELD\_VARIABLE\_TYPE), pointer RESIDUAL\_VARIABLE  
*A pointer to the variable that is mapped to the residual vector for nonlinear problems.*
- TYPE(DOMAIN\_MAPPING\_TYPE), pointer RESIDUAL\_VARIABLE\_MAPPING  
*A pointer to the residual variable domain mapping.*
- REAL(DP) RESIDUAL\_COEFFICIENT  
*The multiplicative coefficient applied to the residual vector.*
- INTEGER(INTG), allocatable RESIDUAL\_DOF\_TO\_EQUATIONS\_ROW\_MAP  
*RESIDUAL\_DOF\_TO\_EQUATIONS\_ROW\_MAP(residual\_dof\_idx). The mapping from the residual\_dof\_-idx'th residual dof in the residual variable to the equations row.*
- INTEGER(INTG), allocatable EQUATIONS\_ROW\_TO\_RESIDUAL\_DOF\_MAP  
*EQUATIONS\_ROW\_TO\_RESIDUAL\_DOF\_MAP(row\_idx). The mapping from the row\_idx'th row of the equations to the source dof.*

### 7.213.1 Detailed Description

Definition at line 1106 of file types.f90.

### 7.213.2 Member Data Documentation

#### 7.213.2.1 TYPE(EQUATIONS\_MAPPING\_TYPE),pointer TYPES::EQUATIONS\_MAPPING\_- NONLINEAR\_TYPE::EQUATIONS\_MAPPING

A pointer to the equations mapping.

Definition at line 1107 of file types.f90.

**7.213.2.2 INTEGER(INTG),allocatable TYPES::EQUATIONS\_MAPPING\_NONLINEAR\_-  
TYPE::EQUATIONS\_ROW\_TO\_RESIDUAL\_DOF\_MAP**

EQUATIONS\_ROW\_TO\_RESIDUAL\_DOF\_MAP(row\_idx). The mapping from the row\_idx'th row of the equations to the source dof.

Definition at line 1115 of file types.f90.

**7.213.2.3 TYPE(EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP\_TYPE)  
TYPES::EQUATIONS\_MAPPING\_NONLINEAR\_TYPE::JACOBIAN\_TO\_-  
VARIABLE\_MAP**

Definition at line 1109 of file types.f90.

**7.213.2.4 REAL(DP) TYPES::EQUATIONS\_MAPPING\_NONLINEAR\_TYPE::RESIDUAL\_-  
COEFFICIENT**

The multiplicative coefficient applied to the residual vector.

Definition at line 1113 of file types.f90.

**7.213.2.5 INTEGER(INTG),allocatable TYPES::EQUATIONS\_MAPPING\_NONLINEAR\_-  
TYPE::RESIDUAL\_DOF\_TO\_EQUATIONS\_ROW\_MAP**

RESIDUAL\_DOF\_TO\_EQUATIONS\_ROW\_MAP(residual\_dof\_idx). The mapping from the residual\_dof\_idx'th residual dof in the residual variable to the equations row.

Definition at line 1114 of file types.f90.

**7.213.2.6 TYPE(FIELD\_VARIABLE\_TYPE),pointer TYPES::EQUATIONS\_MAPPING\_-  
NONLINEAR\_TYPE::RESIDUAL\_VARIABLE**

A pointer to the variable that is mapped to the residual vector for nonlinear problems.

Definition at line 1111 of file types.f90.

**7.213.2.7 TYPE(DOMAIN\_MAPPING\_TYPE),pointer TYPES::EQUATIONS\_MAPPING\_-  
NONLINEAR\_TYPE::RESIDUAL\_VARIABLE\_MAPPING**

A pointer to the residual variable domain mapping.

Definition at line 1112 of file types.f90.

**7.213.2.8 INTEGER(INTG) TYPES::EQUATIONS\_MAPPING\_NONLINEAR\_-  
TYPE::RESIDUAL\_VARIABLE\_TYPE**

Definition at line 1110 of file types.f90.

**7.213.2.9 TYPE(VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_MAP\_TYPE)**  
**TYPES::EQUATIONS\_MAPPING\_NONLINEAR\_TYPE::VARIABLE\_TO\_-**  
**JACOBIAN\_MAP**

Definition at line 1108 of file types.f90.

## 7.214 TYPES::EQUATIONS\_MAPPING\_RHS\_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS\_MAPPING\_RHS\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_MAPPING\_TYPE), pointer EQUATIONS\_MAPPING  
*A pointer to the equations mapping.*
- INTEGER(INTG) RHS\_VARIABLE\_TYPE  
*The variable type number mapped to the RHS vector.*
- TYPE(FIELD\_VARIABLE\_TYPE), pointer RHS\_VARIABLE  
*A pointer to the variable that is mapped to the RHS vector.*
- TYPE(DOMAIN\_MAPPING\_TYPE), pointer RHS\_VARIABLE\_MAPPING  
*A pointer to the RHS variable domain mapping.*
- REAL(DP) RHS\_COEFFICIENT  
*The multiplicative coefficient applied to the RHS vector.*
- INTEGER(INTG), allocatable RHS\_DOF\_TO\_EQUATIONS\_ROW\_MAP  
 $RHS\_DOF\_TO\_EQUATIONS\_ROW\_MAP(residual\_dof\_idx)$ . *The mapping from the rhs\_dof\_idx'th RHS dof in the rhs variable to the equations row.*
- INTEGER(INTG), allocatable EQUATIONS\_ROW\_TO\_RHS\_DOF\_MAP  
 $EQUATIONS\_ROW\_TO\_RHS\_DOF\_MAP(row\_idx)$ . *The mapping from the row\_idx'th row of the equations to the RHS dof.*

### 7.214.1 Detailed Description

Definition at line 1118 of file types.f90.

### 7.214.2 Member Data Documentation

#### 7.214.2.1 TYPE(EQUATIONS\_MAPPING\_TYPE),pointer TYPES::EQUATIONS\_MAPPING\_- RHS\_TYPE::EQUATIONS\_MAPPING

A pointer to the equations mapping.

Definition at line 1119 of file types.f90.

#### 7.214.2.2 INTEGER(INTG),allocatable TYPES::EQUATIONS\_MAPPING\_RHS\_- TYPE::EQUATIONS\_ROW\_TO\_RHS\_DOF\_MAP

$EQUATIONS\_ROW\_TO\_RHS\_DOF\_MAP(row\_idx)$ . The mapping from the row\_idx'th row of the equations to the RHS dof.

Definition at line 1125 of file types.f90.

#### **7.214.2.3 REAL(DP) TYPES::EQUATIONS\_MAPPING\_RHS\_TYPE::RHS\_COEFFICIENT**

The multiplicative coefficient applied to the RHS vector.

Definition at line 1123 of file types.f90.

#### **7.214.2.4 INTEGER(INTG),allocatable TYPES::EQUATIONS\_MAPPING\_RHS\_TYPE::RHS\_- DOF\_TO\_EQUATIONS\_ROW\_MAP**

RHS\_DOF\_TO\_EQUATIONS\_ROW\_MAP(residual\_dof\_idx). The mapping from the rhs\_dof\_idx'th RHS dof in the rhs variable to the equations row.

Definition at line 1124 of file types.f90.

#### **7.214.2.5 TYPE(FIELD\_VARIABLE\_TYPE),pointer TYPES::EQUATIONS\_MAPPING\_RHS\_- TYPE::RHS\_VARIABLE**

A pointer to the variable that is mapped to the RHS vector.

Definition at line 1121 of file types.f90.

#### **7.214.2.6 TYPE(DOMAIN\_MAPPING\_TYPE),pointer TYPES::EQUATIONS\_MAPPING\_- RHS\_TYPE::RHS\_VARIABLE\_MAPPING**

A pointer to the RHS variable domain mapping.

Definition at line 1122 of file types.f90.

#### **7.214.2.7 INTEGER(INTG) TYPES::EQUATIONS\_MAPPING\_RHS\_TYPE::RHS\_- VARIABLE\_TYPE**

The variable type number mapped to the RHS vector.

Definition at line 1120 of file types.f90.

## 7.215 TYPES::EQUATIONS\_MAPPING\_SOURCE\_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS\_MAPPING\_SOURCE\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_MAPPING\_TYPE), pointer EQUATIONS\_MAPPING  
*A pointer to the equations mapping.*
- INTEGER(INTG) SOURCE\_VARIABLE\_TYPE  
*The variable type number mapped from the source vector.*
- TYPE(FIELD\_VARIABLE\_TYPE), pointer SOURCE\_VARIABLE  
*A pointer to the source variable.*
- TYPE(DOMAIN\_MAPPING\_TYPE), pointer SOURCE\_VARIABLE\_MAPPING  
*A pointer to the domain mapping for the source variable.*
- REAL(DP) SOURCE\_COEFFICIENT  
*The multiplicative coefficient applied to the source vector.*
- INTEGER(INTG), allocatable SOURCE\_DOF\_TO\_EQUATIONS\_ROW\_MAP  
 $SOURCE\_DOF\_TO\_EQUATIONS\_ROW\_MAP(source\_dof\_idx)$ . *The mapping from the source\_dof\_idx'th source dof in the source variable to the equations row.*
- INTEGER(INTG), allocatable EQUATIONS\_ROW\_TO\_SOURCE\_DOF\_MAP  
 $EQUATIONS\_ROW\_TO\_SOURCE\_DOF\_MAP(row\_idx)$ . *The mapping from the row\_idx'th row of the equations to the source dof.*

### 7.215.1 Detailed Description

Definition at line 1128 of file types.f90.

### 7.215.2 Member Data Documentation

#### 7.215.2.1 TYPE(EQUATIONS\_MAPPING\_TYPE),pointer TYPES::EQUATIONS\_MAPPING\_SOURCE\_TYPE::EQUATIONS\_MAPPING

A pointer to the equations mapping.

Definition at line 1129 of file types.f90.

#### 7.215.2.2 INTEGER(INTG),allocatable TYPES::EQUATIONS\_MAPPING\_SOURCE\_TYPE::EQUATIONS\_ROW\_TO\_SOURCE\_DOF\_MAP

$EQUATIONS\_ROW\_TO\_SOURCE\_DOF\_MAP(row\_idx)$ . The mapping from the row\_idx'th row of the equations to the source dof.

Definition at line 1135 of file types.f90.

#### **7.215.2.3 REAL(DP) TYPES::EQUATIONS\_MAPPING\_SOURCE\_TYPE::SOURCE\_COEFFICIENT**

The multiplicative coefficient applied to the source vector.

Definition at line 1133 of file types.f90.

#### **7.215.2.4 INTEGER(INTG),allocatable TYPES::EQUATIONS\_MAPPING\_SOURCE\_TYPE::SOURCE\_DOF\_TO\_EQUATIONS\_ROW\_MAP**

SOURCE\_DOF\_TO\_EQUATIONS\_ROW\_MAP(source\_dof\_idx). The mapping from the source\_dof\_idx'th source dof in the source variable to the equations row.

Definition at line 1134 of file types.f90.

#### **7.215.2.5 TYPE(FIELD\_VARIABLE\_TYPE),pointer TYPES::EQUATIONS\_MAPPING\_SOURCE\_TYPE::SOURCE\_VARIABLE**

A pointer to the source variable.

Definition at line 1131 of file types.f90.

#### **7.215.2.6 TYPE(DOMAIN\_MAPPING\_TYPE),pointer TYPES::EQUATIONS\_MAPPING\_SOURCE\_TYPE::SOURCE\_VARIABLE\_MAPPING**

A pointer to the domain mapping for the source variable.

Definition at line 1132 of file types.f90.

#### **7.215.2.7 INTEGER(INTG) TYPES::EQUATIONS\_MAPPING\_SOURCE\_TYPE::SOURCE\_VARIABLE\_TYPE**

The variable type number mapped from the source vector.

Definition at line 1130 of file types.f90.

## 7.216 TYPES::EQUATIONS\_MAPPING\_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS\_MAPPING\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_TYPE), pointer EQUATIONS  
*A pointer to the equations for this equations mapping.*
- LOGICAL EQUATIONS\_MAPPING\_FINISHED  
*Is .TRUE. if the equations mapping has finished being created, .FALSE. if not.*
- TYPE(EQUATIONS\_MATRICES\_TYPE), pointer EQUATIONS\_MATRICES  
*A pointer to the equations matrices associated with this equations mapping.*
- INTEGER(INTG) NUMBER\_OF\_ROWS  
*The number of local rows (excluding ghost rows) in the equations matrices.*
- INTEGER(INTG) TOTAL\_NUMBER\_OF\_ROWS  
*The number of local rows (including ghost rows) in the equations matrices.*
- INTEGER(INTG) NUMBER\_OF\_GLOBAL\_ROWS  
*The number of global rows in the equations matrices.*
- INTEGER(INTG) RESIDUAL\_VARIABLE\_TYPE  
*The variable type number mapped to the residual vector for nonlinear problems.*
- TYPE(FIELD\_VARIABLE\_TYPE), pointer RESIDUAL\_VARIABLE  
*A pointer to the variable that is mapped to the residual vector for nonlinear problems.*
- TYPE(DOMAIN\_MAPPING\_TYPE), pointer RESIDUAL\_VARIABLE\_MAPPING  
*A pointer to the residual variable domain mapping.*
- TYPE(DOMAIN\_MAPPING\_TYPE), pointer ROW\_DOFS\_MAPPING  
*The domain mapping for the equations rows.*
  - TYPE(EQUATIONS\_MAPPING\_LINEAR\_TYPE), pointer LINEAR\_MAPPING
  - TYPE(EQUATIONS\_MAPPING\_NONLINEAR\_TYPE), pointer NONLINEAR\_MAPPING
  - TYPE(EQUATIONS\_MAPPING\_RHS\_TYPE), pointer RHS\_MAPPING
  - TYPE(EQUATIONS\_MAPPING\_SOURCE\_TYPE), pointer SOURCE\_MAPPING
  - TYPE(EQUATIONS\_MAPPING\_CREATE\_VALUES\_CACHE\_TYPE), pointer CREATE\_VALUES\_CACHE  
*The create values cache for the equations mapping.*

### 7.216.1 Detailed Description

Definition at line 1150 of file types.f90.

## 7.216.2 Member Data Documentation

### 7.216.2.1 **TYPE(EQUATIONS\_MAPPING\_CREATE\_VALUES\_CACHE\_TYPE),pointer TYPES::EQUATIONS\_MAPPING\_TYPE::CREATE\_VALUES\_CACHE**

The create values cache for the equations mapping.

Definition at line 1168 of file types.f90.

### 7.216.2.2 **TYPE(EQUATIONS\_TYPE),pointer TYPES::EQUATIONS\_MAPPING\_- TYPE::EQUATIONS**

A pointer to the equations for this equations mapping.

Definition at line 1151 of file types.f90.

### 7.216.2.3 **LOGICAL TYPES::EQUATIONS\_MAPPING\_TYPE::EQUATIONS\_MAPPING\_- FINISHED**

Is .TRUE. if the equations mapping has finished being created, .FALSE. if not.

Definition at line 1152 of file types.f90.

### 7.216.2.4 **TYPE(EQUATIONS\_MATRICES\_TYPE),pointer TYPES::EQUATIONS\_MAPPING\_- TYPE::EQUATIONS\_MATRICES**

A pointer to the equations matrices associated with this equations mapping.

Definition at line 1153 of file types.f90.

### 7.216.2.5 **TYPE(EQUATIONS\_MAPPING\_LINEAR\_TYPE),pointer TYPES::EQUATIONS\_MAPPING\_TYPE::LINEAR\_MAPPING**

Definition at line 1163 of file types.f90.

### 7.216.2.6 **TYPE(EQUATIONS\_MAPPING\_NONLINEAR\_TYPE),pointer TYPES::EQUATIONS\_MAPPING\_TYPE::NONLINEAR\_MAPPING**

Definition at line 1164 of file types.f90.

### 7.216.2.7 **INTEGER(INTG) TYPES::EQUATIONS\_MAPPING\_TYPE::NUMBER\_OF\_- GLOBAL\_ROWS**

The number of global rows in the equations matrices.

Definition at line 1157 of file types.f90.

### 7.216.2.8 **INTEGER(INTG) TYPES::EQUATIONS\_MAPPING\_TYPE::NUMBER\_OF\_ROWS**

The number of local rows (excluding ghost rows) in the equations matrices.

Definition at line 1155 of file types.f90.

**7.216.2.9 TYPE(FIELD\_VARIABLE\_TYPE),pointer TYPES::EQUATIONS\_MAPPING\_-  
TYPE::RESIDUAL\_VARIABLE**

A pointer to the variable that is mapped to the residual vector for nonlinear problems.

Definition at line 1159 of file types.f90.

**7.216.2.10 TYPE(DOMAIN\_MAPPING\_TYPE),pointer TYPES::EQUATIONS\_MAPPING\_-  
TYPE::RESIDUAL\_VARIABLE\_MAPPING**

A pointer to the residual variable domain mapping.

Definition at line 1160 of file types.f90.

**7.216.2.11 INTEGER(INTG) TYPES::EQUATIONS\_MAPPING\_TYPE::RESIDUAL\_-  
VARIABLE\_TYPE**

The variable type number mapped to the residual vector for nonlinear problems.

Definition at line 1158 of file types.f90.

**7.216.2.12 TYPE(EQUATIONS\_MAPPING\_RHS\_TYPE),pointer TYPES::EQUATIONS\_-  
MAPPING\_TYPE::RHS\_MAPPING**

Definition at line 1165 of file types.f90.

**7.216.2.13 TYPE(DOMAIN\_MAPPING\_TYPE),pointer TYPES::EQUATIONS\_MAPPING\_-  
TYPE::ROW\_DOFS\_MAPPING**

The domain mapping for the equations rows.

Definition at line 1161 of file types.f90.

**7.216.2.14 TYPE(EQUATIONS\_MAPPING\_SOURCE\_TYPE),pointer  
TYPES::EQUATIONS\_MAPPING\_TYPE::SOURCE\_MAPPING**

Definition at line 1166 of file types.f90.

**7.216.2.15 INTEGER(INTG) TYPES::EQUATIONS\_MAPPING\_TYPE::TOTAL\_NUMBER\_-  
OF\_ROWS**

The number of local rows (including ghost rows) in the equations matrices.

Definition at line 1156 of file types.f90.

## 7.217 TYPES::EQUATIONS\_MATRICES\_LINEAR\_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS\_MATRICES\_LINEAR\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_MATRICES\_TYPE), pointer EQUATIONS\_MATRICES
- INTEGER(INTG) NUMBER\_OF\_LINEAR\_MATRICES
  - The number of equations matrices defined for the problem.*
- TYPE(EQUATIONS\_MATRIX\_PTR\_TYPE), allocatable MATRICES
  - MATRICES(matrix\_idx)PTR contains the information on the matrix\_id.*

### 7.217.1 Detailed Description

Definition at line 1000 of file types.f90.

### 7.217.2 Member Data Documentation

#### 7.217.2.1 TYPE(EQUATIONS\_MATRICES\_TYPE),pointer TYPES::EQUATIONS\_- MATRICES\_LINEAR\_TYPE::EQUATIONS\_MATRICES

Definition at line 1001 of file types.f90.

#### 7.217.2.2 TYPE(EQUATIONS\_MATRIX\_PTR\_TYPE),allocatable TYPES::EQUATIONS\_- MATRICES\_LINEAR\_TYPE::MATRICES

MATRICES(matrix\_idx)PTR contains the information on the matrix\_id.

Definition at line 1003 of file types.f90.

#### 7.217.2.3 INTEGER(INTG) TYPES::EQUATIONS\_MATRICES\_LINEAR\_TYPE::NUMBER\_- OF\_LINEAR\_MATRICES

The number of equations matrices defined for the problem.

Definition at line 1002 of file types.f90.

## 7.218 TYPES::EQUATIONS\_MATRICES\_NONLINEAR\_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS\_MATRICES\_NONLINEAR\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_MATRICES\_TYPE), pointer EQUATIONS\_MATRICES
- TYPE(EQUATIONS\_JACOBIAN\_TYPE), pointer JACOBIAN
  - A pointer to the Jacobian matrix for nonlinear equations.*
- LOGICAL UPDATE\_RESIDUAL
  - Is .TRUE. if the equations rhs vector is to be updated.*
- TYPE(DISTRIBUTED\_VECTOR\_TYPE), pointer RESIDUAL
  - A pointer to the distributed residual vector for nonlinear equations.*
- TYPE(ELEMENT\_VECTOR\_TYPE) ELEMENT\_RESIDUAL
  - The element residual information for nonlinear equations. Old CMISS name RE1.*

### 7.218.1 Detailed Description

Definition at line 1006 of file types.f90.

### 7.218.2 Member Data Documentation

#### 7.218.2.1 TYPE(ELEMENT\_VECTOR\_TYPE) TYPES::EQUATIONS\_MATRICES\_- NONLINEAR\_TYPE::ELEMENT\_RESIDUAL

The element residual information for nonlinear equations. Old CMISS name RE1.

Definition at line 1011 of file types.f90.

#### 7.218.2.2 TYPE(EQUATIONS\_MATRICES\_TYPE),pointer TYPES::EQUATIONS\_- MATRICES\_NONLINEAR\_TYPE::EQUATIONS\_MATRICES

Definition at line 1007 of file types.f90.

#### 7.218.2.3 TYPE(EQUATIONS\_JACOBIAN\_TYPE),pointer TYPES::EQUATIONS\_- MATRICES\_NONLINEAR\_TYPE::JACOBIAN

A pointer to the Jacobian matrix for nonlinear equations.

Definition at line 1008 of file types.f90.

**7.218.2.4 TYPE(DISTRIBUTED\_VECTOR\_TYPE),pointer TYPES::EQUATIONS\_-  
MATRICES\_NONLINEAR\_TYPE::RESIDUAL**

A pointer to the distributed residual vector for nonlinear equations.

Definition at line 1010 of file types.f90.

**7.218.2.5 LOGICAL TYPES::EQUATIONS\_MATRICES\_NONLINEAR\_TYPE::UPDATE\_-  
RESIDUAL**

Is .TRUE. if the equations rhs vector is to be updated.

Definition at line 1009 of file types.f90.

## 7.219 TYPES::EQUATIONS\_MATRICES\_RHS\_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS\_MATRICES\_RHS\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_MATRICES\_TYPE), pointer EQUATIONS\_MATRICES
- LOGICAL UPDATE\_VECTOR  
*Is .TRUE. if the equations rhs vector is to be updated.*
- TYPE(DISTRIBUTED\_VECTOR\_TYPE), pointer VECTOR  
*A pointer to the distributed global rhs vector data.*
- TYPE(ELEMENT\_VECTOR\_TYPE) ELEMENT\_VECTOR  
*The element rhs information.*

### 7.219.1 Detailed Description

Definition at line 1014 of file types.f90.

### 7.219.2 Member Data Documentation

#### 7.219.2.1 TYPE(ELEMENT\_VECTOR\_TYPE) TYPES::EQUATIONS\_MATRICES\_RHS\_- TYPE::ELEMENT\_VECTOR

The element rhs information.

Definition at line 1018 of file types.f90.

#### 7.219.2.2 TYPE(EQUATIONS\_MATRICES\_TYPE),pointer TYPES::EQUATIONS\_- MATRICES\_RHS\_TYPE::EQUATIONS\_MATRICES

Definition at line 1015 of file types.f90.

#### 7.219.2.3 LOGICAL TYPES::EQUATIONS\_MATRICES\_RHS\_TYPE::UPDATE\_VECTOR

Is .TRUE. if the equations rhs vector is to be updated.

Definition at line 1016 of file types.f90.

#### 7.219.2.4 TYPE(DISTRIBUTED\_VECTOR\_TYPE),pointer TYPES::EQUATIONS\_- MATRICES\_RHS\_TYPE::VECTOR

A pointer to the distributed global rhs vector data.

#### Todo

rename this RHS\_VECTOR

Definition at line 1017 of file types.f90.

## 7.220 TYPES::EQUATIONS\_MATRICES\_SOURCE\_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS\_MATRICES\_SOURCE\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_MATRICES\_TYPE), pointer EQUATIONS\_MATRICES
- LOGICAL UPDATE\_VECTOR  
*Is .TRUE. if the equations rhs vector is to be updated.*
- TYPE(DISTRIBUTED\_VECTOR\_TYPE), pointer VECTOR  
*A pointer to the distributed source vector data.*
- TYPE(ELEMENT\_VECTOR\_TYPE) ELEMENT\_VECTOR  
*The element source information.*

### 7.220.1 Detailed Description

Definition at line 1021 of file types.f90.

### 7.220.2 Member Data Documentation

#### 7.220.2.1 TYPE(ELEMENT\_VECTOR\_TYPE) TYPES::EQUATIONS\_MATRICES\_SOURCE\_- TYPE::ELEMENT\_VECTOR

The element source information.

Definition at line 1025 of file types.f90.

#### 7.220.2.2 TYPE(EQUATIONS\_MATRICES\_TYPE),pointer TYPES::EQUATIONS\_- MATRICES\_SOURCE\_TYPE::EQUATIONS\_MATRICES

Definition at line 1022 of file types.f90.

#### 7.220.2.3 LOGICAL TYPES::EQUATIONS\_MATRICES\_SOURCE\_TYPE::UPDATE\_VECTOR

Is .TRUE. if the equations rhs vector is to be updated.

Definition at line 1023 of file types.f90.

#### 7.220.2.4 TYPE(DISTRIBUTED\_VECTOR\_TYPE),pointer TYPES::EQUATIONS\_- MATRICES\_SOURCE\_TYPE::VECTOR

A pointer to the distributed source vector data.

#### Todo

rename this SOURCE\_VECTOR

Definition at line 1024 of file types.f90.

## 7.221 TYPES::EQUATIONS\_MATRICES\_TYPE Struct Reference

Contains information on the equations matrices and rhs vector.

Collaboration diagram for TYPES::EQUATIONS\_MATRICES\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_TYPE), pointer EQUATIONS  
*A pointer back to the equations.*
- LOGICAL EQUATIONS\_MATRICES\_FINISHED  
*Is .TRUE. if the equations matrices have finished being created, .FALSE. if not.*
- TYPE(EQUATIONS\_MAPPING\_TYPE), pointer EQUATIONS\_MAPPING  
*A pointer to the equations mapping for the equations matrices.*
- TYPE(SOLUTION\_MAPPING\_TYPE), pointer SOLUTION\_MAPPING  
*A pointer to the solution mapping for the equations matrices.*
- INTEGER(INTG) NUMBER\_OF\_ROWS  
*The number of local rows (excluding ghost rows) in the distributed equations matrices and vectors.*
- INTEGER(INTG) TOTAL\_NUMBER\_OF\_ROWS  
*The number of local rows (including ghost rows) in the distributed equations matrices and vectors.*
- INTEGER(INTG) NUMBER\_OF\_GLOBAL\_ROWS  
*The number of global rows in the distributed equations matrices and vectors.*
- TYPE(EQUATIONS\_MATRICES\_LINEAR\_TYPE), pointer LINEAR\_MATRICES
- TYPE(EQUATIONS\_MATRICES\_NONLINEAR\_TYPE), pointer NONLINEAR\_MATRICES
- TYPE(EQUATIONS\_MATRICES\_RHS\_TYPE), pointer RHS\_VECTOR
- TYPE(EQUATIONS\_MATRICES\_SOURCE\_TYPE), pointer SOURCE\_VECTOR

### 7.221.1 Detailed Description

Contains information on the equations matrices and rhs vector.

Definition at line 1029 of file types.f90.

### 7.221.2 Member Data Documentation

#### 7.221.2.1 TYPE(EQUATIONS\_TYPE),pointer TYPES::EQUATIONS\_MATRICES\_- TYPE::EQUATIONS

A pointer back to the equations.

Definition at line 1030 of file types.f90.

**7.221.2.2 TYPE(EQUATIONS\_MAPPING\_TYPE),pointer TYPES::EQUATIONS\_MATRICES\_-  
TYPE::EQUATIONS\_MAPPING**

A pointer to the equations mapping for the equations matrices.

Definition at line 1032 of file types.f90.

**7.221.2.3 LOGICAL TYPES::EQUATIONS\_MATRICES\_TYPE::EQUATIONS\_MATRICES\_-  
FINISHED**

Is .TRUE. if the equations matrices have finished being created, .FALSE. if not.

Definition at line 1031 of file types.f90.

**7.221.2.4 TYPE(EQUATIONS\_MATRICES\_LINEAR\_TYPE),pointer  
TYPES::EQUATIONS\_MATRICES\_TYPE::LINEAR\_MATRICES**

Definition at line 1038 of file types.f90.

**7.221.2.5 TYPE(EQUATIONS\_MATRICES\_NONLINEAR\_TYPE),pointer  
TYPES::EQUATIONS\_MATRICES\_TYPE::NONLINEAR\_MATRICES**

Definition at line 1039 of file types.f90.

**7.221.2.6 INTEGER(INTG) TYPES::EQUATIONS\_MATRICES\_TYPE::NUMBER\_OF\_-  
GLOBAL\_ROWS**

The number of global rows in the distributed equations matrices and vectors.

Definition at line 1036 of file types.f90.

**7.221.2.7 INTEGER(INTG) TYPES::EQUATIONS\_MATRICES\_TYPE::NUMBER\_OF\_ROWS**

The number of local rows (excluding ghost rows) in the distributed equations matrices and vectors.

Definition at line 1034 of file types.f90.

**7.221.2.8 TYPE(EQUATIONS\_MATRICES\_RHS\_TYPE),pointer TYPES::EQUATIONS\_-  
MATRICES\_TYPE::RHS\_VECTOR**

Definition at line 1040 of file types.f90.

**7.221.2.9 TYPE(SOLUTION\_MAPPING\_TYPE),pointer TYPES::EQUATIONS\_MATRICES\_-  
TYPE::SOLUTION\_MAPPING**

A pointer to the solution mapping for the equations matrices.

Definition at line 1033 of file types.f90.

**7.221.2.10 TYPE(EQUATIONS\_MATRICES\_SOURCE\_TYPE),pointer  
TYPES::EQUATIONS\_MATRICES\_TYPE::SOURCE\_VECTOR**

Definition at line 1041 of file types.f90.

**7.221.2.11 INTEGER(INTG) TYPES::EQUATIONS\_MATRICES\_TYPE::TOTAL\_NUMBER\_OF\_ROWS**

The number of local rows (including ghost rows) in the distributed equations matrices and vectors.

Definition at line 1035 of file types.f90.

## 7.222 TYPES::EQUATIONS\_MATRIX\_PTR\_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS\_MATRIX\_PTR\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_MATRIX\_TYPE), pointer PTR

#### 7.222.1 Detailed Description

Definition at line 983 of file types.f90.

#### 7.222.2 Member Data Documentation

##### 7.222.2.1 TYPE(EQUATIONS\_MATRIX\_TYPE),pointer TYPES::EQUATIONS\_MATRIX\_- PTR\_TYPE::PTR

Definition at line 984 of file types.f90.

## 7.223 TYPES::EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_TYPE:

### Public Attributes

- INTEGER(INTG) [MATRIX\\_NUMBER](#)  
*The equations matrix number.*
- TYPE([EQUATIONS\\_MATRIX\\_TYPE](#)), pointer [EQUATIONS\\_MATRIX](#)  
*A pointer to the equations matrix.*
- INTEGER(INTG) [VARIABLE\\_TYPE](#)  
*The dependent variable type mapped to this equations matrix.*
- TYPE([FIELD\\_VARIABLE\\_TYPE](#)), pointer [VARIABLE](#)  
*A pointer to the field variable that is mapped to this equations matrix.*
- INTEGER(INTG) [NUMBER\\_OF\\_COLUMNS](#)  
*The number of columns in this equations matrix.*
- REAL(DP) [MATRIX\\_COEFFICIENT](#)  
*The multiplicative coefficient for the matrix in the equation set.*
- INTEGER(INTG), allocatable [COLUMN\\_TO\\_DOF\\_MAP](#)  
*COLUMN\_TO\_DOF\_MAP(column\_idx). The variable DOF that the column\_idx'th column of this equations matrix is mapped to.*
- TYPE([DOMAIN\\_MAPPING\\_TYPE](#)), pointer [COLUMN\\_DOFS\\_MAPPING](#)  
*A pointer to the column dofs domain mapping for the matrix variable.*

### 7.223.1 Detailed Description

Definition at line 1066 of file types.f90.

### 7.223.2 Member Data Documentation

#### 7.223.2.1 TYPE(DOMAIN\_MAPPING\_TYPE),pointer TYPES::EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_TYPE::COLUMN\_DOFS\_MAPPING

A pointer to the column dofs domain mapping for the matrix variable.

Definition at line 1074 of file types.f90.

**7.223.2.2 INTEGER(INTG),allocatable TYPES::EQUATIONS\_MATRIX\_TO\_VARIABLE\_-  
MAP\_TYPE::COLUMN\_TO\_DOF\_MAP**

COLUMN\_TO\_DOF\_MAP(column\_idx). The variable DOF that the column\_idx'th column of this equations matrix is mapped to.

Definition at line 1073 of file types.f90.

**7.223.2.3 TYPE(EQUATIONS\_MATRIX\_TYPE),pointer TYPES::EQUATIONS\_MATRIX\_TO\_-  
VARIABLE\_MAP\_TYPE::EQUATIONS\_MATRIX**

A pointer to the equations matrix.

Definition at line 1068 of file types.f90.

**7.223.2.4 REAL(DP) TYPES::EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_-  
TYPE::MATRIX\_COEFFICIENT**

The multiplicative coefficient for the matrix in the equation set.

Definition at line 1072 of file types.f90.

**7.223.2.5 INTEGER(INTG) TYPES::EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_-  
TYPE::MATRIX\_NUMBER**

The equations matrix number.

Definition at line 1067 of file types.f90.

**7.223.2.6 INTEGER(INTG) TYPES::EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_-  
TYPE::NUMBER\_OF\_COLUMNS**

The number of columns in this equations matrix.

Definition at line 1071 of file types.f90.

**7.223.2.7 TYPE(FIELD\_VARIABLE\_TYPE),pointer TYPES::EQUATIONS\_MATRIX\_TO\_-  
VARIABLE\_MAP\_TYPE::VARIABLE**

A pointer to the field variable that is mapped to this equations matrix.

Definition at line 1070 of file types.f90.

**7.223.2.8 INTEGER(INTG) TYPES::EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_-  
TYPE::VARIABLE\_TYPE**

The dependent variable type mapped to this equations matrix.

Definition at line 1069 of file types.f90.

## 7.224 TYPES::EQUATIONS\_MATRIX\_TYPE Struct Reference

Contains information about an equations matrix.

Collaboration diagram for TYPES::EQUATIONS\_MATRIX\_TYPE:

### Public Attributes

- INTEGER(INTG) [MATRIX\\_NUMBER](#)  
*The number of the equations matrix.*
- TYPE([EQUATIONS\\_MATRICES\\_LINEAR\\_TYPE](#)), pointer [LINEAR\\_MATRICES](#)  
*A pointer to the equations matrices for the equation matrix.*
- INTEGER(INTG) [STORAGE\\_TYPE](#)  
*The storage (sparsity) type for this matrix.*
- INTEGER(INTG) [STRUCTURE\\_TYPE](#)  
*The structure (sparsity) type for this matrix.*
- INTEGER(INTG) [NUMBER\\_OF\\_COLUMNS](#)  
*The number of columns in this global matrix.*
- LOGICAL [UPDATE\\_MATRIX](#)  
*Is .TRUE. if this global matrix is to be updated.*
- TYPE([DISTRIBUTED\\_MATRIX\\_TYPE](#)), pointer [MATRIX](#)  
*A pointer to the distributed global matrix data.*
- TYPE([ELEMENT\\_MATRIX\\_TYPE](#)) [ELEMENT\\_MATRIX](#)  
*The element matrix for this glboal matrix.*

### 7.224.1 Detailed Description

Contains information about an equations matrix.

Definition at line 972 of file types.f90.

### 7.224.2 Member Data Documentation

#### 7.224.2.1 TYPE(ELEMENT\_MATRIX\_TYPE) TYPES::EQUATIONS\_MATRIX\_- TYPE::ELEMENT\_MATRIX

The element matrix for this glboal matrix.

Definition at line 980 of file types.f90.

**7.224.2.2 TYPE(EQUATIONS\_MATRICES\_LINEAR\_TYPE),pointer  
TYPES::EQUATIONS\_MATRIX\_TYPE::LINEAR\_MATRICES**

A pointer to the equations matrices for the equation matrix.

Definition at line 974 of file types.f90.

**7.224.2.3 TYPE(DISTRIBUTED\_MATRIX\_TYPE),pointer TYPES::EQUATIONS\_MATRIX\_-  
TYPE::MATRIX**

A pointer to the distributed global matrix data.

Definition at line 979 of file types.f90.

**7.224.2.4 INTEGER(INTG) TYPES::EQUATIONS\_MATRIX\_TYPE::MATRIX\_NUMBER**

The number of the equations matrix.

Definition at line 973 of file types.f90.

**7.224.2.5 INTEGER(INTG) TYPES::EQUATIONS\_MATRIX\_TYPE::NUMBER\_OF\_-  
COLUMNS**

The number of columns in this global matrix.

Definition at line 977 of file types.f90.

**7.224.2.6 INTEGER(INTG) TYPES::EQUATIONS\_MATRIX\_TYPE::STORAGE\_TYPE**

The storage (sparsity) type for this matrix.

Definition at line 975 of file types.f90.

**7.224.2.7 INTEGER(INTG) TYPES::EQUATIONS\_MATRIX\_TYPE::STRUCTURE\_TYPE**

The structure (sparsity) type for this matrix.

Definition at line 976 of file types.f90.

**7.224.2.8 LOGICAL TYPES::EQUATIONS\_MATRIX\_TYPE::UPDATE\_MATRIX**

Is .TRUE. if this global matrix is to be updated.

Definition at line 978 of file types.f90.

## 7.225 TYPES::EQUATIONS\_NONLINEAR\_DATA\_TYPE Struct Reference

Contains information on any data required for a non-linear solution.

Collaboration diagram for TYPES::EQUATIONS\_NONLINEAR\_DATA\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_TYPE), pointer EQUATIONS

*A pointer to the equations.*

#### 7.225.1 Detailed Description

Contains information on any data required for a non-linear solution.

Definition at line 1205 of file types.f90.

#### 7.225.2 Member Data Documentation

##### 7.225.2.1 TYPE(EQUATIONS\_TYPE),pointer TYPES::EQUATIONS\_NONLINEAR\_DATA\_TYPE::EQUATIONS

A pointer to the equations.

Definition at line 1206 of file types.f90.

## 7.226 TYPES::EQUATIONS\_ROW\_TO\_SOLVER\_ROWS\_MAP\_- TYPE Struct Reference

Contains information on the mapping from the equations rows in an equations set to the solver rows.

### Public Attributes

- INTEGER(INTG) **NUMBER\_OF\_SOLVER\_ROWS**  
*The number of solver rows this equations row is mapped to.*
- INTEGER(INTG), allocatable **SOLVER\_ROWS**  
*SOLVER\_ROWS(i). The solver row number for the i'th solver row that this row is mapped to.*
- REAL(DP), allocatable **COUPLING\_COEFFICIENTS**  
*COUPLING\_COEFFICIENTS(i). The coupling coefficients for the i'th solver row that this row is mapped to.*

### 7.226.1 Detailed Description

Contains information on the mapping from the equations rows in an equations set to the solver rows.

Definition at line 1535 of file types.f90.

### 7.226.2 Member Data Documentation

#### 7.226.2.1 REAL(DP),allocatable TYPES::EQUATIONS\_ROW\_TO\_SOLVER\_ROWS\_MAP\_- TYPE::COUPLING\_COEFFICIENTS

COUPLING\_COEFFICIENTS(i). The coupling coefficients for the i'th solver row that this row is mapped to.

Definition at line 1538 of file types.f90.

#### 7.226.2.2 INTEGER(INTG) TYPES::EQUATIONS\_ROW\_TO\_SOLVER\_ROWS\_MAP\_- TYPE::NUMBER\_OF\_SOLVER\_ROWS

The number of solver rows this equations row is mapped to.

Definition at line 1536 of file types.f90.

#### 7.226.2.3 INTEGER(INTG),allocatable TYPES::EQUATIONS\_ROW\_TO\_SOLVER\_ROWS\_- MAP\_TYPE::SOLVER\_ROWS

SOLVER\_ROWS(i). The solver row number for the i'th solver row that this row is mapped to.

Definition at line 1537 of file types.f90.

## 7.227 TYPES::EQUATIONS\_SET\_ANALYTIC\_TYPE Struct Reference

Contains information on the analytic setup for the equations set.

Collaboration diagram for TYPES::EQUATIONS\_SET\_ANALYTIC\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_SET\_TYPE), pointer EQUATIONS\_SET  
*A pointer to the equations set.*
- INTEGER(INTG) EQUATION\_NUMBER  
*The equation identifier.*
- LOGICAL ANALYTIC\_FINISHED  
*Is .TRUE. if the analytic setup for the problem has finished being created, .FALSE. if not.*

### 7.227.1 Detailed Description

Contains information on the analytic setup for the equations set.

Definition at line 1275 of file types.f90.

### 7.227.2 Member Data Documentation

#### 7.227.2.1 LOGICAL TYPES::EQUATIONS\_SET\_ANALYTIC\_TYPE::ANALYTIC\_FINISHED

Is .TRUE. if the analytic setup for the problem has finished being created, .FALSE. if not.

Definition at line 1278 of file types.f90.

#### 7.227.2.2 INTEGER(INTG) TYPES::EQUATIONS\_SET\_ANALYTIC\_TYPE::EQUATION\_NUMBER

The equation identifier.

Definition at line 1277 of file types.f90.

#### 7.227.2.3 TYPE(EQUATIONS\_SET\_TYPE),pointer TYPES::EQUATIONS\_SET\_ANALYTIC\_TYPE::EQUATIONS\_SET

A pointer to the equations set.

Definition at line 1276 of file types.f90.

## 7.228 TYPES::EQUATIONS\_SET\_DEPENDENT\_TYPE Struct Reference

Contains information on the dependent variables for the equations set.

Collaboration diagram for TYPES::EQUATIONS\_SET\_DEPENDENT\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_SET\_TYPE), pointer EQUATIONS\_SET  
*A pointer to the equations set.*
- LOGICAL DEPENDENT\_FINISHED  
*Is .TRUE. if the dependent variables for the equations set has finished being created, .FALSE. if not.*
- TYPE(FIELD\_TYPE), pointer DEPENDENT\_FIELD  
*A pointer to the dependent field for the equations set.*

### 7.228.1 Detailed Description

Contains information on the dependent variables for the equations set.

Definition at line 1259 of file types.f90.

### 7.228.2 Member Data Documentation

#### 7.228.2.1 TYPE(FIELD\_TYPE),pointer TYPES::EQUATIONS\_SET\_DEPENDENT\_- TYPE::DEPENDENT\_FIELD

A pointer to the dependent field for the equations set.

Definition at line 1262 of file types.f90.

#### 7.228.2.2 LOGICAL TYPES::EQUATIONS\_SET\_DEPENDENT\_TYPE::DEPENDENT\_- FINISHED

Is .TRUE. if the dependent variables for the equations set has finished being created, .FALSE. if not.

Definition at line 1261 of file types.f90.

#### 7.228.2.3 TYPE(EQUATIONS\_SET\_TYPE),pointer TYPES::EQUATIONS\_SET\_- DEPENDENT\_TYPE::EQUATIONS\_SET

A pointer to the equations set.

Definition at line 1260 of file types.f90.

## 7.229 TYPES::EQUATIONS\_SET\_FIXED\_CONDITIONS\_TYPE Struct Reference

Contains information on the fixed conditions for the problem.

Collaboration diagram for TYPES::EQUATIONS\_SET\_FIXED\_CONDITIONS\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_SET\_TYPE), pointer EQUATIONS\_SET  
*A pointer to the equations set.*
- LOGICAL FIXED\_CONDITIONS\_FINISHED  
*Is .TRUE. if the fixed conditions for the equations set has finished being created, .FALSE. if not.*
- INTEGER(INTG), allocatable GLOBAL\_BOUNDARY\_CONDITIONS  
*GLOBAL\_BOUNDARY\_CONDITIONS(dof\_idx). The global boundary condition for the dof\_idx'th dof of the dependent field.*
- TYPE(DISTRIBUTED\_VECTOR\_TYPE), pointer BOUNDARY\_CONDITIONS  
*A pointer to the distributed vector containing the boundary conditions for the domain for this process.*

### 7.229.1 Detailed Description

Contains information on the fixed conditions for the problem.

Definition at line 1251 of file types.f90.

### 7.229.2 Member Data Documentation

#### 7.229.2.1 TYPE(DISTRIBUTED\_VECTOR\_TYPE),pointer TYPES::EQUATIONS\_SET\_FIXED\_CONDITIONS\_TYPE::BOUNDARY\_CONDITIONS

A pointer to the distributed vector containing the boundary conditions for the domain for this process.

Definition at line 1255 of file types.f90.

#### 7.229.2.2 TYPE(EQUATIONS\_SET\_TYPE),pointer TYPES::EQUATIONS\_SET\_FIXED\_CONDITIONS\_TYPE::EQUATIONS\_SET

A pointer to the equations set.

Definition at line 1252 of file types.f90.

#### 7.229.2.3 LOGICAL TYPES::EQUATIONS\_SET\_FIXED\_CONDITIONS\_TYPE::FIXED\_CONDITIONS\_FINISHED

Is .TRUE. if the fixed conditions for the equations set has finished being created, .FALSE. if not.

Definition at line 1253 of file types.f90.

**7.229.2.4 INTEGER(INTG),allocatable TYPES::EQUATIONS\_SET\_FIXED\_CONDITIONS\_-  
TYPE::GLOBAL\_BOUNDARY\_CONDITIONS**

GLOBAL\_BOUNDARY\_CONDITIONS(dof\_idx). The global boundary condition for the dof\_idx'th dof of the dependent field.

**See also:**

EQUATIONS\_ROUTINES\_FixedConditions,EQUATIONS\_ROUTINES

Definition at line 1254 of file types.f90.

## 7.230 TYPES::EQUATIONS\_SET\_GEOMETRY\_TYPE Struct Reference

Contains information on the geometry for an equations set.

Collaboration diagram for TYPES::EQUATIONS\_SET\_GEOMETRY\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_SET\_TYPE), pointer EQUATIONS\_SET  
*A pointer to the equations set.*
- TYPE(FIELD\_TYPE), pointer GEOMETRIC\_FIELD  
*The geometric field for this equations set.*
- TYPE(FIELD\_TYPE), pointer FIBRE\_FIELD  
*The fibre field for this equations set if one is defined. If no fibre field is defined the pointer is NULL.*

### 7.230.1 Detailed Description

Contains information on the geometry for an equations set.

Definition at line 1238 of file types.f90.

### 7.230.2 Member Data Documentation

#### 7.230.2.1 TYPE(EQUATIONS\_SET\_TYPE),pointer TYPES::EQUATIONS\_SET\_GEOMETRY\_- TYPE::EQUATIONS\_SET

A pointer to the equations set.

Definition at line 1239 of file types.f90.

#### 7.230.2.2 TYPE(FIELD\_TYPE),pointer TYPES::EQUATIONS\_SET\_GEOMETRY\_- TYPE::FIBRE\_FIELD

The fibre field for this equations set if one is defined. If no fibre field is defined the pointer is NULL.

Definition at line 1241 of file types.f90.

#### 7.230.2.3 TYPE(FIELD\_TYPE),pointer TYPES::EQUATIONS\_SET\_GEOMETRY\_- TYPE::GEOMETRIC\_FIELD

The geometric field for this equations set.

Definition at line 1240 of file types.f90.

## 7.231 TYPES::EQUATIONS\_SET\_MATERIALS\_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS\_SET\_MATERIALS\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_SET\_TYPE), pointer EQUATIONS\_SET  
*A pointer to the equations set.*
- LOGICAL MATERIALS\_FINISHED  
*Is .TRUE. if the materials for the equations set has finished being created, .FALSE. if not.*
- TYPE(FIELD\_TYPE), pointer MATERIAL\_FIELD  
*A pointer to the material field for the equations set if one is defined. If no material field is defined the pointer is NULL.*

### 7.231.1 Detailed Description

Definition at line 1244 of file types.f90.

### 7.231.2 Member Data Documentation

#### 7.231.2.1 TYPE(EQUATIONS\_SET\_TYPE),pointer TYPES::EQUATIONS\_SET\_- MATERIALS\_TYPE::EQUATIONS\_SET

A pointer to the equations set.

Definition at line 1245 of file types.f90.

#### 7.231.2.2 TYPE(FIELD\_TYPE),pointer TYPES::EQUATIONS\_SET\_MATERIALS\_- TYPE::MATERIAL\_FIELD

A pointer to the material field for the equations set if one is defined. If no material field is defined the pointer is NULL.

Definition at line 1247 of file types.f90.

#### 7.231.2.3 LOGICAL TYPES::EQUATIONS\_SET\_MATERIALS\_TYPE::MATERIALS\_- FINISHED

Is .TRUE. if the materials for the equations set has finished being created, .FALSE. if not.

Definition at line 1246 of file types.f90.

## 7.232 TYPES::EQUATIONS\_SET\_PTR\_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS\_SET\_PTR\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_SET\_TYPE), pointer PTR

#### 7.232.1 Detailed Description

Definition at line 1308 of file types.f90.

#### 7.232.2 Member Data Documentation

##### 7.232.2.1 TYPE(EQUATIONS\_SET\_TYPE),pointer TYPES::EQUATIONS\_SET\_PTR\_- TYPE::PTR

Definition at line 1309 of file types.f90.

## 7.233 TYPES::EQUATIONS\_SET\_SOURCE\_TYPE Struct Reference

Contains information on the source for the equations set.

Collaboration diagram for TYPES::EQUATIONS\_SET\_SOURCE\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_SET\_TYPE), pointer EQUATIONS\_SET  
*A pointer to the equations set.*
- LOGICAL SOURCE\_FINISHED  
*Is .TRUE. if the source for the equations set has finished being created, .FALSE. if not.*
- TYPE(FIELD\_TYPE), pointer SOURCE\_FIELD  
*A pointer to the source field for the equations set if one is defined. If no source is defined the pointer is NULL.*

### 7.233.1 Detailed Description

Contains information on the source for the equations set.

Definition at line 1266 of file types.f90.

### 7.233.2 Member Data Documentation

#### 7.233.2.1 TYPE(EQUATIONS\_SET\_TYPE),pointer TYPES::EQUATIONS\_SET\_SOURCE\_TYPE::EQUATIONS\_SET

A pointer to the equations set.

Definition at line 1267 of file types.f90.

#### 7.233.2.2 TYPE(FIELD\_TYPE),pointer TYPES::EQUATIONS\_SET\_SOURCE\_TYPE::SOURCE\_FIELD

A pointer to the source field for the equations set if one is defined. If no source is defined the pointer is NULL.

Definition at line 1269 of file types.f90.

#### 7.233.2.3 LOGICAL TYPES::EQUATIONS\_SET\_SOURCE\_TYPE::SOURCE\_FINISHED

Is .TRUE. if the source for the equations set has finished being created, .FALSE. if not.

Definition at line 1268 of file types.f90.

## 7.234 TYPES::EQUATIONS\_SET\_TO\_SOLVER\_MAP\_TYPE Struct Reference

Contains information on the mappings from an equations set to the solver matrices.

Collaboration diagram for TYPES::EQUATIONS\_SET\_TO\_SOLVER\_MAP\_TYPE:

### Public Attributes

- INTEGER(INTG) **EQUATIONS\_SET\_INDEX**  
*The index of the equations set for these mappings.*
- TYPE(**SOLUTION\_MAPPING\_TYPE**), pointer **SOLUTION\_MAPPING**  
*A pointer to the solution mappings.*
- TYPE(**EQUATIONS\_TYPE**), pointer **EQUATIONS**  
*A pointer to the equations in this equations set.*
- TYPE(**EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_SM\_TYPE**), allocatable **EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_SM**  
*EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_SM(solver\_matrix\_idx). The mappings from the equations matrices in this equation set to the solver\_matrix\_idx'th solver\_matrix.*
- TYPE(**EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_EM\_TYPE**), allocatable **EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_EM**  
*EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_EM(equations\_matrix\_idx). The mappings from the equation\_matrix\_idx'th equations matrix in this equation set to the solver\_matrices.*
- TYPE(**EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_JM\_TYPE**), pointer **EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_JM**  
*The mappings from the Jacobian matrix in this equation set to the solver\_matrices.*
- TYPE(**EQUATIONS\_ROW\_TO\_SOLVER\_ROWS\_MAP\_TYPE**), allocatable **EQUATIONS\_ROW\_TO\_SOLVER\_ROWS\_MAPS**  
*EQUATIONS\_ROW\_TO\_SOLVER\_ROWS\_MAPS(equations\_row\_idx). The mappings from the equations\_row\_idx'th equations row to the solver matrices rows.*

### 7.234.1 Detailed Description

Contains information on the mappings from an equations set to the solver matrices.

Definition at line 1542 of file types.f90.

### 7.234.2 Member Data Documentation

#### 7.234.2.1 TYPE(**EQUATIONS\_TYPE**),pointer TYPES::EQUATIONS\_SET\_TO\_SOLVER\_MAP\_TYPE::**EQUATIONS**

A pointer to the equations in this equations set.

Definition at line 1545 of file types.f90.

**7.234.2.2 TYPE(EQUATIONS\_ROW\_TO\_SOLVER\_ROWS\_MAP\_TYPE),allocatable  
TYPES::EQUATIONS\_SET\_TO\_SOLVER\_MAP\_TYPE::EQUATIONS\_ROW\_TO\_-  
SOLVER\_ROWS\_MAPS**

EQUATIONS\_ROW\_TO\_SOLVER\_ROWS\_MAPS(equations\_row\_idx). The mappings from the equations\_row\_idx'th equations row to the solver matrices rows.

Definition at line 1549 of file types.f90.

**7.234.2.3 INTEGER(INTG) TYPES::EQUATIONS\_SET\_TO\_SOLVER\_MAP\_-  
TYPE::EQUATIONS\_SET\_INDEX**

The index of the equations set for these mappings.

Definition at line 1543 of file types.f90.

**7.234.2.4 TYPE(EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_EM\_TYPE),allocatable  
TYPES::EQUATIONS\_SET\_TO\_SOLVER\_MAP\_TYPE::EQUATIONS\_TO\_-  
SOLVER\_MATRIX\_MAPS\_EM**

EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_EM(equations\_matrix\_idx). The mappings from the equation\_matrix\_idx'th equations matrix in this equation set to the solver\_matrices.

Definition at line 1547 of file types.f90.

**7.234.2.5 TYPE(EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_JM\_TYPE),pointer  
TYPES::EQUATIONS\_SET\_TO\_SOLVER\_MAP\_TYPE::EQUATIONS\_TO\_-  
SOLVER\_MATRIX\_MAPS\_JM**

The mappings from the Jacobian matrix in this equation set to the solver\_matrices.

Definition at line 1548 of file types.f90.

**7.234.2.6 TYPE(EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_SM\_TYPE),allocatable  
TYPES::EQUATIONS\_SET\_TO\_SOLVER\_MAP\_TYPE::EQUATIONS\_TO\_-  
SOLVER\_MATRIX\_MAPS\_SM**

EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_SM(solver\_matrix\_idx). The mappings from the equations matrices in this equation set to the solver\_matrix\_idx'th solver\_matrix.

Definition at line 1546 of file types.f90.

**7.234.2.7 TYPE(SOLUTION\_MAPPING\_TYPE),pointer TYPES::EQUATIONS\_SET\_TO\_-  
SOLVER\_MAP\_TYPE::SOLUTION\_MAPPING**

A pointer to the solution mappings.

Definition at line 1544 of file types.f90.

## 7.235 TYPES::EQUATIONS\_SET\_TYPE Struct Reference

Contains information on an equations set.

Collaboration diagram for TYPES::EQUATIONS\_SET\_TYPE:

### Public Attributes

- INTEGER(INTG) **USER\_NUMBER**  
*The user identifying number of the equations set.*
- INTEGER(INTG) **GLOBAL\_NUMBER**  
*The global index of the equations set in the region.*
- LOGICAL **EQUATIONS\_SET\_FINISHED**  
*Is .TRUE. if the equations set have finished being created, .FALSE. if not.*
- TYPE(EQUATIONS\_SETS\_TYPE), pointer **EQUATIONS\_SETS**  
*A pointer back to the equations sets.*
- TYPE(REGION\_TYPE), pointer **REGION**  
*A pointer back to the region containing the equations set.*
- INTEGER(INTG) **CLASS**  
*The equations set specification class identifier.*
- INTEGER(INTG) **TYPE**  
*The equations set specification type identifier.*
- INTEGER(INTG) **SUBTYPE**  
*The equations set specification subtype identifier.*
- INTEGER(INTG) **LINEARITY**  
*The equations set linearity type.*
- INTEGER(INTG) **TIME\_TYPE**  
*The equations set time dependence type.*
- INTEGER(INTG) **SOLUTION\_METHOD**  
*The solution method for the equations set.*
- TYPE(EQUATIONS\_SET\_GEOMETRY\_TYPE) **GEOMETRY**  
*The geometry information for the equations set.*
- TYPE(EQUATIONS\_SET\_MATERIALS\_TYPE), pointer **MATERIALS**  
*A pointer to the materials information for the equations set.*
- TYPE(EQUATIONS\_SET\_SOURCE\_TYPE), pointer **SOURCE**  
*A pointer to the source information for the equations set.*

- **TYPE(EQUATIONS\_SET\_DEPENDENT\_TYPE) DEPENDENT**

*The depedent variable information for the equations set.*

- **TYPE(EQUATIONS\_SET\_ANALYTIC\_TYPE), pointer ANALYTIC**

*A pointer to the analytic setup information for the equations set.*

- **TYPE(EQUATIONS\_SET\_FIXED\_CONDITIONS\_TYPE), pointer FIXED\_CONDITIONS**

*A pointer to the fixed condition information for the equations set.*

- **TYPE(EQUATIONS\_TYPE), pointer EQUATIONS**

### 7.235.1 Detailed Description

Contains information on an equations set.

Definition at line 1282 of file types.f90.

### 7.235.2 Member Data Documentation

#### 7.235.2.1 TYPE(EQUATIONS\_SET\_ANALYTIC\_TYPE),pointer TYPES::EQUATIONS\_SET\_- TYPE::ANALYTIC

A pointer to the analytic setup information for the equations set.

Definition at line 1303 of file types.f90.

#### 7.235.2.2 INTEGER(INTG) TYPES::EQUATIONS\_SET\_TYPE::CLASS

The equations set specification class identifier.

Definition at line 1289 of file types.f90.

#### 7.235.2.3 TYPE(EQUATIONS\_SET\_DEPENDENT\_TYPE) TYPES::EQUATIONS\_SET\_- TYPE::DEPENDENT

The depedent variable information for the equations set.

Definition at line 1302 of file types.f90.

#### 7.235.2.4 TYPE(EQUATIONS\_TYPE),pointer TYPES::EQUATIONS\_SET\_TYPE::EQUATIONS

Definition at line 1305 of file types.f90.

#### 7.235.2.5 LOGICAL TYPES::EQUATIONS\_SET\_TYPE::EQUATIONS\_SET\_FINISHED

Is .TRUE. if the equations set have finished being created, .FALSE. if not.

Definition at line 1285 of file types.f90.

**7.235.2.6 TYPE(EQUATIONS\_SETS\_TYPE),pointer TYPES::EQUATIONS\_SET\_-  
TYPE::EQUATIONS\_SETS**

A pointer back to the equations sets.

Definition at line 1286 of file types.f90.

**7.235.2.7 TYPE(EQUATIONS\_SET\_FIXED\_CONDITIONS\_TYPE),pointer  
TYPES::EQUATIONS\_SET\_TYPE::FIXED\_CONDITIONS**

A pointer to the fixed condition information for the equations set.

**Todo**

Change name to BOUNDARY\_CONDITIONS???

Definition at line 1304 of file types.f90.

**7.235.2.8 TYPE(EQUATIONS\_SET\_GEOMETRY\_TYPE) TYPES::EQUATIONS\_SET\_-  
TYPE::GEOMETRY**

The geometry information for the equations set.

Definition at line 1299 of file types.f90.

**7.235.2.9 INTEGER(INTG) TYPES::EQUATIONS\_SET\_TYPE::GLOBAL\_NUMBER**

The global index of the equations set in the region.

Definition at line 1284 of file types.f90.

**7.235.2.10 INTEGER(INTG) TYPES::EQUATIONS\_SET\_TYPE::LINEARITY**

The equations set linearity type.

**See also:**

EQUATIONS\_ROUTINES\_LinearityTypes

Definition at line 1294 of file types.f90.

**7.235.2.11 TYPE(EQUATIONS\_SET\_MATERIALS\_TYPE),pointer  
TYPES::EQUATIONS\_SET\_TYPE::MATERIALS**

A pointer to the materials information for the equations set.

Definition at line 1300 of file types.f90.

**7.235.2.12 TYPE(REGION\_TYPE),pointer TYPES::EQUATIONS\_SET\_TYPE::REGION**

A pointer back to the region containing the equations set.

Definition at line 1287 of file types.f90.

**7.235.2.13 INTEGER(INTG) TYPES::EQUATIONS\_SET\_TYPE::SOLUTION\_METHOD**

The solution method for the equations set.

**See also:**

EQUATIONS\_ROUTINES\_SolutionMethods

Definition at line 1297 of file types.f90.

**7.235.2.14 TYPE(EQUATIONS\_SET\_SOURCE\_TYPE),pointer TYPES::EQUATIONS\_SET\_TYPE::SOURCE**

A pointer to the source information for the equations set.

Definition at line 1301 of file types.f90.

**7.235.2.15 INTEGER(INTG) TYPES::EQUATIONS\_SET\_TYPE::SUBTYPE**

The equations set specification subtype identifier.

Definition at line 1291 of file types.f90.

**7.235.2.16 INTEGER(INTG) TYPES::EQUATIONS\_SET\_TYPE::TIME\_TYPE**

The equations set time dependence type.

**See also:**

EQUATIONS\_ROUTINES\_TimeDepedenceTypes

Definition at line 1295 of file types.f90.

**7.235.2.17 INTEGER(INTG) TYPES::EQUATIONS\_SET\_TYPE::TYPE**

The equations set specification type identifier.

Definition at line 1290 of file types.f90.

**7.235.2.18 INTEGER(INTG) TYPES::EQUATIONS\_SET\_TYPE::USER\_NUMBER**

The user identifying number of the equations set.

Definition at line 1283 of file types.f90.

## 7.236 TYPES::EQUATIONS\_SETS\_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS\_SETS\_TYPE:

### Public Attributes

- TYPE(REGION\_TYPE), pointer REGION
- INTEGER(INTG) NUMBER\_OF\_EQUATIONS\_SETS
- TYPE(EQUATIONS\_SET\_PTR\_TYPE), pointer EQUATIONS\_SETS

#### 7.236.1 Detailed Description

Definition at line 1312 of file types.f90.

#### 7.236.2 Member Data Documentation

##### 7.236.2.1 TYPE(EQUATIONS\_SET\_PTR\_TYPE),pointer TYPES::EQUATIONS\_SETS\_- TYPE::EQUATIONS\_SETS

Definition at line 1315 of file types.f90.

##### 7.236.2.2 INTEGER(INTG) TYPES::EQUATIONS\_SETS\_TYPE::NUMBER\_OF\_- EQUATIONS\_SETS

Definition at line 1314 of file types.f90.

##### 7.236.2.3 TYPE(REGION\_TYPE),pointer TYPES::EQUATIONS\_SETS\_TYPE::REGION

Definition at line 1313 of file types.f90.

## 7.237 TYPES::EQUATIONS\_TIME\_DATA\_TYPE Struct Reference

Contains information on any data required for a time-dependent equations.

Collaboration diagram for TYPES::EQUATIONS\_TIME\_DATA\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_TYPE), pointer EQUATIONS

*A pointer to the equations.*

#### 7.237.1 Detailed Description

Contains information on any data required for a time-dependent equations.

Definition at line 1210 of file types.f90.

#### 7.237.2 Member Data Documentation

##### 7.237.2.1 TYPE(EQUATIONS\_TYPE),pointer TYPES::EQUATIONS\_TIME\_DATA\_-TYPE::EQUATIONS

A pointer to the equations.

Definition at line 1211 of file types.f90.

## 7.238 TYPES::EQUATIONS\_TO\_SOLVER\_MAPS\_PTR\_TYPE Struct Reference

A buffer type to allow for an array of pointers to a EQUATIONS\_TO\_SOLVER\_MAPS\_TYPE.

Collaboration diagram for TYPES::EQUATIONS\_TO\_SOLVER\_MAPS\_PTR\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_TO\_SOLVER\_MAPS\_TYPE), pointer PTR  
*A pointer to the equations to solver maps.*

#### 7.238.1 Detailed Description

A buffer type to allow for an array of pointers to a EQUATIONS\_TO\_SOLVER\_MAPS\_TYPE.

See also:

[TYPES::EQUATIONS\\_TO\\_SOLVER\\_MAPS\\_TYPE](#)

Definition at line 1486 of file types.f90.

#### 7.238.2 Member Data Documentation

##### 7.238.2.1 TYPE(EQUATIONS\_TO\_SOLVER\_MAPS\_TYPE),pointer TYPES::EQUATIONS\_TO\_SOLVER\_MAPS\_PTR\_TYPE::PTR

A pointer to the equations to solver maps.

Definition at line 1487 of file types.f90.

## 7.239 TYPES::EQUATIONS\_TO\_SOLVER\_MAPS\_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS\_TO\_SOLVER\_MAPS\_TYPE:

### Public Attributes

- INTEGER(INTG) [EQUATIONS\\_MATRIX\\_NUMBER](#)  
*The equations matrix number being mapped.*
- INTEGER(INTG) [SOLVER\\_MATRIX\\_NUMBER](#)  
*The solver matrix number being mapped.*
- TYPE([EQUATIONS\\_MATRIX\\_TYPE](#)), pointer [EQUATIONS\\_MATRIX](#)  
*A pointer to the equations matrix being mapped.*
- TYPE([SOLVER\\_MATRIX\\_TYPE](#)), pointer [SOLVER\\_MATRIX](#)  
*A pointer to the solver matrix being mapped.*
- TYPE([EQUATIONS\\_COL\\_TO\\_SOLVER\\_COLS\\_MAP\\_TYPE](#)), allocatable [EQUATIONS\\_COL\\_-  
SOLVER\\_COLS\\_MAP](#)  
*EQUATIONS\_COL\_SOLVER\_COL\_MAP(column\_idx). The mapping from the column\_idx'th column of  
this equations matrix to the solver matrix columns.*

### 7.239.1 Detailed Description

Definition at line 1477 of file types.f90.

### 7.239.2 Member Data Documentation

#### 7.239.2.1 TYPE([EQUATIONS\\_COL\\_TO\\_SOLVER\\_COLS\\_MAP\\_TYPE](#)),allocatable TYPES::EQUATIONS\_TO\_SOLVER\_MAPS\_TYPE::[EQUATIONS\\_COL\\_SOLVER\\_- COLS\\_MAP](#)

EQUATIONS\_COL\_SOLVER\_COL\_MAP(column\_idx). The mapping from the column\_idx'th column of this equations matrix to the solver matrix columns.

Definition at line 1482 of file types.f90.

#### 7.239.2.2 TYPE([EQUATIONS\\_MATRIX\\_TYPE](#)),pointer TYPES::EQUATIONS\_TO\_SOLVER\_- MAPS\_TYPE::[EQUATIONS\\_MATRIX](#)

A pointer to the equations matrix being mapped.

Definition at line 1480 of file types.f90.

**7.239.2.3 INTEGER(INTG) TYPES::EQUATIONS\_TO\_SOLVER\_MAPS\_-  
TYPE::EQUATIONS\_MATRIX\_NUMBER**

The equations matrix number being mapped.

Definition at line 1478 of file types.f90.

**7.239.2.4 TYPE(SOLVER\_MATRIX\_TYPE),pointer TYPES::EQUATIONS\_TO\_SOLVER\_-  
MAPS\_TYPE::SOLVER\_MATRIX**

A pointer to the solver matrix being mapped.

Definition at line 1481 of file types.f90.

**7.239.2.5 INTEGER(INTG) TYPES::EQUATIONS\_TO\_SOLVER\_MAPS\_TYPE::SOLVER\_-  
MATRIX\_NUMBER**

The solver matrix number being mapped.

Definition at line 1479 of file types.f90.

## 7.240 TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_EM\_TYPE Struct Reference

Contains information on the equations to solver matrix mappings when indexing by equations matrix number.

Collaboration diagram for TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_EM\_TYPE:

### Public Attributes

- INTEGER(INTG) [EQUATIONS\\_MATRIX\\_NUMBER](#)  
*The number of the equations matrix for these mappings.*
- INTEGER(INTG) [NUMBER\\_OF\\_SOLVER\\_MATRICES](#)  
*The number of solver matrices mapped to this equations matrix.*
- TYPE([EQUATIONS\\_TO\\_SOLVER\\_MAPS\\_PTR\\_TYPE](#)), allocatable [EQUATIONS\\_TO\\_SOLVER\\_MATRIX\\_MAPS](#)  
*EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS(solver\_matrix\_idx). The maps from the equation matrix to the solver\_matrix\_idx'th solver matrix.*

### 7.240.1 Detailed Description

Contains information on the equations to solver matrix mappings when indexing by equations matrix number.

Definition at line 1524 of file types.f90.

### 7.240.2 Member Data Documentation

#### 7.240.2.1 INTEGER(INTG) TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_EM\_TYPE::EQUATIONS\_MATRIX\_NUMBER

The number of the equations matrix for these mappings.

Definition at line 1525 of file types.f90.

#### 7.240.2.2 TYPE(EQUATIONS\_TO\_SOLVER\_MAPS\_PTR\_TYPE),allocatable TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_EM\_TYPE::EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS

EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS(solver\_matrix\_idx). The maps from the equation matrix to the solver\_matrix\_idx'th solver matrix.

Definition at line 1527 of file types.f90.

#### 7.240.2.3 INTEGER(INTG) TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_EM\_TYPE::NUMBER\_OF\_SOLVER\_MATRICES

The number of solver matrices mapped to this equations matrix.

Definition at line 1526 of file types.f90.

## 7.241 TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_-JM\_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_JM\_TYPE:

### Public Attributes

- TYPE(JACOBIAN\_TO\_SOLVER\_MAP\_TYPE), pointer JACOBIAN\_TO\_SOLVER\_MATRIX\_-MAP

#### 7.241.1 Detailed Description

Definition at line 1530 of file types.f90.

#### 7.241.2 Member Data Documentation

##### 7.241.2.1 TYPE(JACOBIAN\_TO\_SOLVER\_MAP\_TYPE),pointer TYPES::EQUATIONS\_TO\_-SOLVER\_MATRIX\_MAPS\_JM\_TYPE::JACOBIAN\_TO\_SOLVER\_MATRIX\_MAP

Definition at line 1531 of file types.f90.

## 7.242 TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_SM\_TYPE Struct Reference

Contains information on the equations to solver matrix mappings when indexing by solver matrix number.

Collaboration diagram for TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_SM\_TYPE:

### Public Attributes

- INTEGER(INTG) [SOLVER\\_MATRIX\\_NUMBER](#)

*The number of the solver matrix for these mappings.*

- INTEGER(INTG) [NUMBER\\_OF\\_VARIABLES](#)

*The number of variables mapped to this solver matrix.*

- INTEGER(INTG), allocatable [VARIABLE\\_TYPES](#)

*VARIABLE\_TYPES(variable\_idx). The variable type for the variable\_idx'th variable that is mapped to the solver matrix.*

- TYPE([FIELD\\_VARIABLE\\_PTR\\_TYPE](#)), allocatable [VARIABLES](#)

*VARIABLES(variable\_idx). VARIABLES(variable\_idx)PTR points to the variable\_idx'th variable that is mapped to the solver matrix.*

- TYPE([VARIABLE\\_TO\\_SOLVER\\_COL\\_MAP\\_TYPE](#)), allocatable [VARIABLE\\_TO\\_SOLVER\\_COL\\_MAPS](#)

*VARIABLE\_TO\_SOLVER\_COL\_MAPS(variable\_idx). The mappings from the variable dofs to the solver dofs for the variable\_idx'th variable in the equations set that is mapped to the solver matrix.*

- INTEGER(INTG) [NUMBER\\_OF\\_LINEAR\\_EQUATIONS\\_MATRICES](#)

*The number of linear equations matrices mapped to this solver matrix.*

- TYPE([EQUATIONS\\_TO\\_SOLVER\\_MAPS\\_PTR\\_TYPE](#)), allocatable [EQUATIONS\\_TO\\_SOLVER\\_MATRIX\\_MAPS](#)

*EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS(equations\_matrix\_idx). The maps from the equations\_idx'th linear equations matrix to solver matrix.*

- TYPE([JACOBIAN\\_TO\\_SOLVER\\_MAP\\_TYPE](#)), pointer [JACOBIAN\\_TO\\_SOLVER\\_MATRIX\\_MAP](#)

*The map from the Jacobian matrix to the solver matrix.*

### 7.242.1 Detailed Description

Contains information on the equations to solver matrix mappings when indexing by solver matrix number.

Definition at line 1511 of file types.f90.

## 7.242.2 Member Data Documentation

**7.242.2.1** **TYPE(EQUATIONS\_TO\_SOLVER\_MAPS\_PTR\_TYPE),allocatable  
TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_SM\_TYPE::EQUATIONS\_-  
TO\_SOLVER\_MATRIX\_MAPS**

EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS(equations\_matrix\_idx). The maps from the equations\_-idx'th linear equations matrix to solver matrix.

Definition at line 1519 of file types.f90.

**7.242.2.2** **TYPE(JACOBIAN\_TO\_SOLVER\_MAP\_TYPE),pointer TYPES::EQUATIONS\_TO\_-  
SOLVER\_MATRIX\_MAPS\_SM\_TYPE::JACOBIAN\_TO\_SOLVER\_MATRIX\_MAP**

The map from the Jacobian matrix to the solver matrix.

Definition at line 1520 of file types.f90.

**7.242.2.3** **INTEGER(INTG) TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_SM\_-  
TYPE::NUMBER\_OF\_LINEAR\_EQUATIONS\_MATRICES**

The number of linear equations matrices mapped to this solver matrix.

Definition at line 1518 of file types.f90.

**7.242.2.4** **INTEGER(INTG) TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_SM\_-  
TYPE::NUMBER\_OF\_VARIABLES**

The number of variables mapped to this solver matrix.

Definition at line 1514 of file types.f90.

**7.242.2.5** **INTEGER(INTG) TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_SM\_-  
TYPE::SOLVER\_MATRIX\_NUMBER**

The number of the solver matrix for these mappings.

Definition at line 1512 of file types.f90.

**7.242.2.6** **TYPE(VARIABLE\_TO\_SOLVER\_COL\_MAP\_TYPE),allocatable  
TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_SM\_TYPE::VARIABLE\_-  
TO\_SOLVER\_COL\_MAPS**

VARIABLE\_TO\_SOLVER\_COL\_MAPS(variable\_idx). The mappings from the variable dofs to the solver dofs for the variable\_idx'th variable in the equations set that is mapped to the solver matrix.

Definition at line 1517 of file types.f90.

**7.242.2.7** **INTEGER(INTG),allocatable TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_-  
MAPS\_SM\_TYPE::VARIABLE\_TYPES**

VARIABLE\_TYPES(variable\_idx). The variable type for the variable\_idx'th variable that is mapped to the solver matrix.

Definition at line 1515 of file types.f90.

**7.242.2.8 TYPE(FIELD\_VARIABLE\_PTR\_TYPE),allocatable TYPES::EQUATIONS\_TO\_-  
SOLVER\_MATRIX\_MAPS\_SM\_TYPE::VARIABLES**

VARIABLES(variable\_idx). VARIABLES(variable\_idx)PTR points to the variable\_idx'th variable that is mapped to the solver matrix.

Definition at line 1516 of file types.f90.

## 7.243 TYPES::EQUATIONS\_TYPE Struct Reference

Contains information about the equations in an equations set.

Collaboration diagram for TYPES::EQUATIONS\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_SET\_TYPE), pointer EQUATIONS\_SET  
*A pointer to the equations\_set.*
- LOGICAL EQUATIONS\_FINISHED  
*Is .TRUE. if the equations have finished being created, .FALSE. if not.*
- INTEGER(INTG) OUTPUT\_TYPE  
*The output type for the equations.*
- INTEGER(INTG) SPARSITY\_TYPE  
*The sparsity type for the equation matrices of the equations.*
- INTEGER(INTG) NONLINEAR\_JACOBIAN\_TYPE  
*The type of non-linear Jacobian calculation.*
- TYPE(EQUATIONS\_INTERPOLATION\_TYPE), pointer INTERPOLATION  
*A pointer to the interpolation information used in the equations.*
- TYPE(EQUATIONS\_LINEAR\_DATA\_TYPE), pointer LINEAR\_DATA  
*A pointer to the data for linear equations.*
- TYPE(EQUATIONS\_NONLINEAR\_DATA\_TYPE), pointer NONLINEAR\_DATA  
*A pointer to the data for non-linear equations.*
- TYPE(EQUATIONS\_TIME\_DATA\_TYPE), pointer TIME\_DATA  
*A pointer to the data for non-static equations.*
- TYPE(EQUATIONS\_MAPPING\_TYPE), pointer EQUATIONS\_MAPPING  
*A pointer to the equations mapping for the equations.*
- TYPE(EQUATIONS\_MATRICES\_TYPE), pointer EQUATIONS\_MATRICES  
*A pointer to the equations matrices and vectors used for the equations.*

### 7.243.1 Detailed Description

Contains information about the equations in an equations set.

Definition at line 1215 of file types.f90.

## 7.243.2 Member Data Documentation

### 7.243.2.1 LOGICAL TYPES::EQUATIONS\_TYPE::EQUATIONS\_FINISHED

Is .TRUE. if the equations have finished being created, .FALSE. if not.

Definition at line 1217 of file types.f90.

### 7.243.2.2 TYPE(EQUATIONS\_MAPPING\_TYPE),pointer TYPES::EQUATIONS\_- TYPE::EQUATIONS\_MAPPING

A pointer to the equations mapping for the equations.

Definition at line 1227 of file types.f90.

### 7.243.2.3 TYPE(EQUATIONS\_MATRICES\_TYPE),pointer TYPES::EQUATIONS\_- TYPE::EQUATIONS\_MATRICES

A pointer to the equations matrices and vectors used for the equations.

Definition at line 1228 of file types.f90.

### 7.243.2.4 TYPE(EQUATIONS\_SET\_TYPE),pointer TYPES::EQUATIONS\_- TYPE::EQUATIONS\_SET

A pointer to the equations\_set.

Definition at line 1216 of file types.f90.

### 7.243.2.5 TYPE(EQUATIONS\_INTERPOLATION\_TYPE),pointer TYPES::EQUATIONS\_- TYPE::INTERPOLATION

A pointer to the interpolation information used in the equations.

Definition at line 1221 of file types.f90.

### 7.243.2.6 TYPE(EQUATIONS\_LINEAR\_DATA\_TYPE),pointer TYPES::EQUATIONS\_- TYPE::LINEAR\_DATA

A pointer to the data for linear equations.

Definition at line 1223 of file types.f90.

### 7.243.2.7 TYPE(EQUATIONS\_NONLINEAR\_DATA\_TYPE),pointer TYPES::EQUATIONS\_TYPE::NONLINEAR\_DATA

A pointer to the data for non-linear equations.

Definition at line 1224 of file types.f90.

**7.243.2.8 INTEGER(INTG) TYPES::EQUATIONS\_TYPE::NONLINEAR\_JACOBIAN\_TYPE**

The type of non-linear Jacobian calculation.

**See also:**

EQUATIONS\_SET\_CONSTANTS\_NonlinearJacobianTypes,[EQUATIONS\\_SET\\_CONSTANTS](#)

Definition at line 1220 of file types.f90.

**7.243.2.9 INTEGER(INTG) TYPES::EQUATIONS\_TYPE::OUTPUT\_TYPE**

The output type for the equations.

**See also:**

EQUATIONS\_ROUTINES\_EquationsOutputTypes,EQUATIONS\_ROUTINES

**Todo**

move to equations set???

Definition at line 1218 of file types.f90.

**7.243.2.10 INTEGER(INTG) TYPES::EQUATIONS\_TYPE::SPARSITY\_TYPE**

The sparsity type for the equation matrices of the equations.

**See also:**

EQUATIONS\_ROUTINES\_EquationsSparsityTypes,EQUATIONS\_ROUTINES

**Todo**

move to equations set???

Definition at line 1219 of file types.f90.

**7.243.2.11 TYPE(EQUATIONS\_TIME\_DATA\_TYPE),pointer TYPES::EQUATIONS\_TYPE::TIME\_DATA**

A pointer to the data for non-static equations.

Definition at line 1225 of file types.f90.

## 7.244 TYPES::FIELD\_CREATE\_VALUES\_CACHE\_TYPE Struct Reference

A type to temporarily hold (cache) the user modifiable values which are used to create a field.

### Public Attributes

- INTEGER(INTG) [NUMBER\\_OF\\_COMPONENTS](#)

*The number of components in the field for each field variable. NOTE: in the future this will need a variable index on it but just allow for the same number of components for each field variable for now.*

- INTEGER(INTG), allocatable [VARIABLE\\_TYPES](#)

*VARIABLE\_TYPES(variable\_idx). The cache of the variable type for the given variable\_idx of the field.*

- INTEGER(INTG), allocatable [INTERPOLATION\\_TYPE](#)

*INTERPOLATION\_TYPES(component\_idx,variable\_idx). The cache of the interpolation type for the given component and variable of the field.*

- INTEGER(INTG), allocatable [MESH\\_COMPONENT\\_NUMBER](#)

*MESH\_COMPONENT\_NUMBER(component\_idx,variable\_idx). The cache of the mesh component number for the given component and variable of the field.*

### 7.244.1 Detailed Description

A type to temporarily hold (cache) the user modifiable values which are used to create a field.

Definition at line 877 of file types.f90.

### 7.244.2 Member Data Documentation

#### 7.244.2.1 INTEGER(INTG),allocatable TYPES::FIELD\_CREATE\_VALUES\_CACHE\_TYPE::INTERPOLATION\_TYPE

*INTERPOLATION\_TYPES(component\_idx,variable\_idx). The cache of the interpolation type for the given component and variable of the field.*

See also:

[FIELD\\_ROUTINES::InterpolationTypes](#)

Definition at line 880 of file types.f90.

#### 7.244.2.2 INTEGER(INTG),allocatable TYPES::FIELD\_CREATE\_VALUES\_CACHE\_TYPE::MESH\_COMPONENT\_NUMBER

*MESH\_COMPONENT\_NUMBER(component\_idx,variable\_idx). The cache of the mesh component number for the given component and variable of the field.*

Definition at line 881 of file types.f90.

**7.244.2.3 INTEGER(INTG) TYPES::FIELD\_CREATE\_VALUES\_CACHE\_TYPE::NUMBER\_OF\_COMPONENTS**

The number of components in the field for each field variable. NOTE: in the future this will need a variable index on it but just allow for the same number of components for each field variable for now.

Definition at line 878 of file types.f90.

**7.244.2.4 INTEGER(INTG),allocatable TYPES::FIELD\_CREATE\_VALUES\_CACHE\_TYPE::VARIABLE\_TYPES**

VARIABLE\_TYPES(variable\_idx). The cache of the variable type for the given variable\_idx of the field.

**See also:**

[FIELD\\_ROUTINES::VariableTypes](#)

Definition at line 879 of file types.f90.

## 7.245 TYPES::FIELD\_OF\_TO\_PARAM\_MAP\_TYPE Struct Reference

A type to hold the mapping from field dof numbers to field parameters (nodes, elements, etc).

### Public Attributes

- INTEGER(INTG) [NUMBER\\_OF\\_DOFS](#)

*The number of degrees-of-freedom for the field.*

- INTEGER(INTG), allocatable [DOF\\_TYPE](#)

*DOF\_TYPE( $i=1..2,ny$ ). The parameter type of the  $ny$ 'th dof. When  $i=1$  the DOF\_TYPE is the type of the dof, i.e., 1=constant field dof, 2=element based field dof, 3=node based field dof, 4=point based field dof. When  $i=2$  the DOF\_TYPE gives the  $nyy$ 'th number of the different field types and is used to index the XXX\_OF2PARAM\_MAP arrays.*

- INTEGER(INTG), allocatable [VARIABLE\\_OF](#)

*VARIABLE\_OF( $ny$ ). The variable dof number for the field  $ny$ .*

- INTEGER(INTG) [NUMBER\\_OF\\_CONSTANT\\_DOFS](#)

*The number of constant degrees-of-freedom in the field dofs.*

- INTEGER(INTG) [NUMBER\\_OF\\_ELEMENT\\_DOFS](#)

*The number of element based degrees-of-freedom in the field dofs.*

- INTEGER(INTG) [NUMBER\\_OF\\_NODE\\_DOFS](#)

*The number of node based degrees-of-freedom in the field dofs.*

- INTEGER(INTG) [NUMBER\\_OF\\_POINT\\_DOFS](#)

*The number of point based degrees-of-freedom in the field dofs.*

- INTEGER(INTG), allocatable [CONSTANT\\_OF2PARAM\\_MAP](#)

*CONSTANT\_OF2PARAM\_MAP( $i=1..2,nyy$ ). The mapping from constant field dofs to field parameters for the  $nyy$ 'th constant field dof. When  $i=1$  the DOF2PARAM\_MAP gives the component number ( $nh$ ) of the field parameter. When  $i=2$  the DOF2PARAM\_MAP gives the variable number ( $nc$ ) of the field parameter. The  $nyy$  value for a particular field dof ( $ny$ ) is given by the DOF\_TYPE component of this type.*

- INTEGER(INTG), allocatable [ELEMENT\\_OF2PARAM\\_MAP](#)

*ELEMENT\_OF2PARAM\_MAP( $i=1..3,nyy$ ). The mapping from element based field dofs to field parameters for the  $nyy$ 'th constant field dof. When  $i=1$  the DOF2PARAM\_MAP gives the element number ( $ne$ ) of the field parameter. When  $i=2$  the DOF2PARAM\_MAP gives the component number ( $nh$ ) of the field parameter. When  $i=3$  the DOF2PARAM\_MAP gives the variable number ( $nc$ ) of the field parameter. The  $nyy$  value for a particular field dof ( $ny$ ) is given by the DOF\_TYPE component of this type.*

- INTEGER(INTG), allocatable [NODE\\_OF2PARAM\\_MAP](#)

*NODE\_OF2PARAM\_MAP( $i=1..4,nyy$ ). The mapping from node based field dofs to field parameters for the  $nyy$ 'th constant field dof. When  $i=1$  the DOF2PARAM\_MAP gives the derivative number ( $nk$ ) of the field parameter. When  $i=2$  the DOF2PARAM\_MAP gives the node number ( $np$ ) of the field parameter. When  $i=3$  the DOF2PARAM\_MAP gives the component number ( $nh$ ) of the field parameter. When  $i=4$  the DOF2PARAM\_MAP gives the variable number ( $nc$ ) of the field parameter. The  $nyy$  value for a particular field dof ( $ny$ ) is given by the DOF\_TYPE component of this type.*

- INTEGER(INTG), allocatable **POINT\_OF2PARAM\_MAP**

*POINT\_OF2PARAM\_MAP(i=1..3,nyy). The mapping from point based field dofs to field parameters for the nyy'th constant field dof. When i=1 the DOF2PARAM\_MAP gives the point number (nq) of the field parameter. When i=2 the DOF2PARAM\_MAP gives the component number (nh) of the field parameter. When i=3 the DOF2PARAM\_MAP gives the variable number (nc) of the field parameter. The nyy value for a particular field dof (ny) is given by the DOF\_TYPE component of this type.*

### 7.245.1 Detailed Description

A type to hold the mapping from field dof numbers to field parameters (nodes, elements, etc).

Definition at line 812 of file types.f90.

### 7.245.2 Member Data Documentation

#### 7.245.2.1 INTEGER(INTG),allocatable TYPES::FIELD\_OF\_TO\_PARAM\_MAP\_- TYPE::CONSTANT\_OF2PARAM\_MAP

CONSTANT\_OF2PARAM\_MAP(i=1..2,nyy). The mapping from constant field dofs to field parameters for the nyy'th constant field dof. When i=1 the DOF2PARAM\_MAP gives the component number (nh) of the field parameter. When i=2 the DOF2PARAM\_MAP gives the variable number (nc) of the field parameter. The nyy value for a particular field dof (ny) is given by the DOF\_TYPE component of this type.

Definition at line 820 of file types.f90.

#### 7.245.2.2 INTEGER(INTG),allocatable TYPES::FIELD\_OF\_TO\_PARAM\_MAP\_- TYPE::DOF\_TYPE

DOF\_TYPE(i=1..2,ny). The parameter type of the ny'th dof. When i=1 the DOF\_TYPE is the type of the dof, i.e., 1=constant field dof, 2=element based field dof, 3=node based field dof, 4=point based field dof. When i=2 the DOF\_TYPE gives the nyy'th number of the different field types and is used to index the XXX\_OF2PARAM\_MAP arrays.

Definition at line 814 of file types.f90.

#### 7.245.2.3 INTEGER(INTG),allocatable TYPES::FIELD\_OF\_TO\_PARAM\_MAP\_- TYPE::ELEMENT\_OF2PARAM\_MAP

ELEMENT\_OF2PARAM\_MAP(i=1..3,nyy). The mapping from element based field dofs to field parameters for the nyy'th constant field dof. When i=1 the DOF2PARAM\_MAP gives the element number (ne) of the field parameter. When i=2 the DOF2PARAM\_MAP gives the component number (nh) of the field parameter. When i=3 the DOF2PARAM\_MAP gives the variable number (nc) of the field parameter. The nyy value for a particular field dof (ny) is given by the DOF\_TYPE component of this type.

Definition at line 821 of file types.f90.

**7.245.2.4 INTEGER(INTG),allocatable TYPES::FIELD\_DOF\_TO\_PARAM\_MAP\_-  
TYPE::NODE\_DOF2PARAM\_MAP**

NODE\_DOF2PARAM\_MAP(i=1..4,nyy). The mapping from node based field dofs to field parameters for the nyy'th constant field dof. When i=1 the DOF2PARAM\_MAP gives the derivative number (nk) of the field parameter. When i=2 the DOF2PARAM\_MAP gives the node number (np) of the field parameter. When i=3 the DOF2PARAM\_MAP gives the component number (nh) of the field parameter. When i=4 the DOF2PARAM\_MAP gives the variable number (nc) of the field parameter. The nyy value for a particular field dof (ny) is given by the DOF\_TYPE component of this type.

Definition at line 822 of file types.f90.

**7.245.2.5 INTEGER(INTG) TYPES::FIELD\_DOF\_TO\_PARAM\_MAP\_TYPE::NUMBER\_OF\_-  
CONSTANT\_DOFS**

The number of constant degrees-of-freedom in the field dofs.

Definition at line 816 of file types.f90.

**7.245.2.6 INTEGER(INTG) TYPES::FIELD\_DOF\_TO\_PARAM\_MAP\_TYPE::NUMBER\_OF\_-  
DOFS**

The number of degrees-of-freedom for the field.

Definition at line 813 of file types.f90.

**7.245.2.7 INTEGER(INTG) TYPES::FIELD\_DOF\_TO\_PARAM\_MAP\_TYPE::NUMBER\_OF\_-  
ELEMENT\_DOFS**

The number of element based degrees-of-freedom in the field dofs.

Definition at line 817 of file types.f90.

**7.245.2.8 INTEGER(INTG) TYPES::FIELD\_DOF\_TO\_PARAM\_MAP\_TYPE::NUMBER\_OF\_-  
NODE\_DOFS**

The number of node based degrees-of-freedom in the field dofs.

Definition at line 818 of file types.f90.

**7.245.2.9 INTEGER(INTG) TYPES::FIELD\_DOF\_TO\_PARAM\_MAP\_TYPE::NUMBER\_OF\_-  
POINT\_DOFS**

The number of point based degrees-of-freedom in the field dofs.

Definition at line 819 of file types.f90.

**7.245.2.10 INTEGER(INTG),allocatable TYPES::FIELD\_DOF\_TO\_PARAM\_MAP\_-  
TYPE::POINT\_DOF2PARAM\_MAP**

POINT\_DOF2PARAM\_MAP(i=1..3,nyy). The mapping from point based field dofs to field parameters for the nyy'th constant field dof. When i=1 the DOF2PARAM\_MAP gives the point number (nq) of the field

parameter. When i=2 the DOF2PARAM\_MAP gives the component number (nh) of the field parameter. When i=3 the DOF2PARAM\_MAP gives the variable number (nc) of the field parameter. The nyy value for a particular field dof (ny) is given by the DOF\_TYPE component of this type.

Definition at line 823 of file types.f90.

#### **7.245.2.11 INTEGER(INTG),allocatable TYPES::FIELD\_OF\_TO\_PARAM\_MAP\_- TYPE::VARIABLE\_OF**

VARIABLE\_OF(ny). The variable dof number for the field ny.

Definition at line 815 of file types.f90.

## 7.246 TYPES::FIELD\_GEOMETRIC\_PARAMETERS\_TYPE Struct Reference

Contains the geometric parameters (lines, faces, volumes etc.) for a geometric field decomposition.

Collaboration diagram for TYPES::FIELD\_GEOMETRIC\_PARAMETERS\_TYPE:

### Public Attributes

- INTEGER(INTG) [NUMBER\\_OF\\_LINES](#)  
*The number of lines in the field.*
- INTEGER(INTG) [NUMBER\\_OF AREAS](#)  
*The number of areas in the field.*
- INTEGER(INTG) [NUMBER\\_OF\\_VOLUMES](#)  
*The number of volumes in the field. Inherited from the field decomposition.*
- REAL(DP), allocatable [LENGTHS](#)  
*LENGTHS(nl). The length of the nl'th line in the field decomposition.*
- REAL(DP), allocatable [AREAS](#)  
*AREAS(nf). The area of the nf'th face in the field decomposition.*
- REAL(DP), allocatable [VOLUMES](#)  
*VOLUMES(ne). The volume of the ne'th element in the field decomposition.*
- INTEGER(INTG) [NUMBER\\_OF\\_FIELDS\\_USING](#)  
*The number of fields that use these geometric parameters for their scaling.*
- TYPE(FIELD\_PTR\_TYPE), pointer [FIELDS\\_USING](#)  
*FIELDS\_USINGS(field\_idx). A pointer to the field\_idx'th field that uses these geometric parameters for its scaling.*

### 7.246.1 Detailed Description

Contains the geometric parameters (lines, faces, volumes etc.) for a geometric field decomposition.

Definition at line 785 of file types.f90.

### 7.246.2 Member Data Documentation

#### 7.246.2.1 REAL(DP),allocatable TYPES::FIELD\_GEOMETRIC\_PARAMETERS\_- TYPE::AREAS

AREAS(nf). The area of the nf'th face in the field decomposition.

Definition at line 790 of file types.f90.

**7.246.2.2 TYPE(FIELD\_PTR\_TYPE),pointer TYPES::FIELD\_GEOMETRIC\_PARAMETERS\_-  
TYPE::FIELDS\_USING**

FIELDS\_USINGS(field\_idx). A pointer to the field\_idx'th field that uses these geometric parameters for its scaling.

Definition at line 793 of file types.f90.

**7.246.2.3 REAL(DP),allocatable TYPES::FIELD\_GEOMETRIC\_PARAMETERS\_-  
TYPE::LENGTHS**

LENGTHS(nl). The length of the nl'th line in the field decomposition.

Definition at line 789 of file types.f90.

**7.246.2.4 INTEGER(INTG) TYPES::FIELD\_GEOMETRIC\_PARAMETERS\_-  
TYPE::NUMBER\_OF AREAS**

The number of areas in the field.

Definition at line 787 of file types.f90.

**7.246.2.5 INTEGER(INTG) TYPES::FIELD\_GEOMETRIC\_PARAMETERS\_-  
TYPE::NUMBER\_OF\_FIELDS\_USING**

The number of fields that use these geometric parameters for their scaling.

Definition at line 792 of file types.f90.

**7.246.2.6 INTEGER(INTG) TYPES::FIELD\_GEOMETRIC\_PARAMETERS\_-  
TYPE::NUMBER\_OF\_LINES**

The number of lines in the field.

Definition at line 786 of file types.f90.

**7.246.2.7 INTEGER(INTG) TYPES::FIELD\_GEOMETRIC\_PARAMETERS\_-  
TYPE::NUMBER\_OF\_VOLUMES**

The number of volumes in the field. Inherited from the field decomposition.

Definition at line 788 of file types.f90.

**7.246.2.8 REAL(DP),allocatable TYPES::FIELD\_GEOMETRIC\_PARAMETERS\_-  
TYPE::VOLUMES**

VOLUMES(ne). The volume of the ne'th element in the field decomposition.

Definition at line 791 of file types.f90.

## 7.247 TYPES::FIELD\_INTERPOLATED\_POINT\_METRICS\_- TYPE Struct Reference

Contains the interpolated point coordinate metrics. Old CMISS name GL, GU, RG.

Collaboration diagram for TYPES::FIELD\_INTERPOLATED\_POINT\_METRICS\_TYPE:

### Public Attributes

- TYPE(FIELD\_INTERPOLATED\_POINT\_TYPE), pointer INTERPOLATED\_POINT  
*A pointer to the interpolated point.*
- INTEGER(INTG) NUMBER\_OF\_X\_DIMENSIONS  
*The number of X dimensions.*
- INTEGER(INTG) NUMBER\_OF\_XI\_DIMENSIONS  
*The number of Xi dimensions.*
- REAL(DP), allocatable GL  
*GL(mi,ni). Covariant metric tensor. Old CMISS name GL.*
- REAL(DP), allocatable GU  
*GU(mi,ni). Contravariant metric tensor. Old CMISS name GU.*
- REAL(DP), allocatable DXI\_DXI  
*DXI\_DXI(nj,ni). Rate of change of the X coordinate system wrt the x coordinate system.*
- REAL(DP), allocatable DXI\_DX  
*DXI\_DX(ni,nj). Rate of change of the Xi coordinate system wrt the x coordinate system.*
- REAL(DP) JACOBIAN  
*The Jacobian of the Xi to X coordinate system transformation. Old CMISS name RG.*
- INTEGER(INTG) JACOBIAN\_TYPE  
*The type of Jacobian.*

### 7.247.1 Detailed Description

Contains the interpolated point coordinate metrics. Old CMISS name GL, GU, RG.

Definition at line 755 of file types.f90.

### 7.247.2 Member Data Documentation

#### 7.247.2.1 REAL(DP),allocatable TYPES::FIELD\_INTERPOLATED\_POINT\_METRICS\_- TYPE::DX\_DXI

DXI\_DXI(nj,ni). Rate of change of the X coordinate system wrt the x coordinate system.

Definition at line 761 of file types.f90.

**7.247.2.2 REAL(DP),allocatable TYPES::FIELD\_INTERPOLATED\_POINT\_METRICS\_-  
TYPE::DXI\_DX**

DXI\_DX(ni,nj). Rate of change of the Xi coordinate system wrt the x coordinate system.

Definition at line 762 of file types.f90.

**7.247.2.3 REAL(DP),allocatable TYPES::FIELD\_INTERPOLATED\_POINT\_METRICS\_-  
TYPE::GL**

GL(mi,ni). Covariant metric tensor. Old [CMISS](#) name GL.

Definition at line 759 of file types.f90.

**7.247.2.4 REAL(DP),allocatable TYPES::FIELD\_INTERPOLATED\_POINT\_METRICS\_-  
TYPE::GU**

GU(mi,ni). Contravariant metric tensor. Old [CMISS](#) name GU.

Definition at line 760 of file types.f90.

**7.247.2.5 TYPE(FIELD\_INTERPOLATED\_POINT\_TYPE),pointer TYPES::FIELD\_-  
INTERPOLATED\_POINT\_METRICS\_TYPE::INTERPOLATED\_POINT**

A pointer to the interpolated point.

Definition at line 756 of file types.f90.

**7.247.2.6 REAL(DP) TYPES::FIELD\_INTERPOLATED\_POINT\_METRICS\_-  
TYPE::JACOBIAN**

The Jacobian of the Xi to X coordinate system transformation. Old [CMISS](#) name RG.

Definition at line 763 of file types.f90.

**7.247.2.7 INTEGER(INTG) TYPES::FIELD\_INTERPOLATED\_POINT\_METRICS\_-  
TYPE::JACOBIAN\_TYPE**

The type of Jacobian.

**See also:**

[COORDINATE\\_ROUTINES::JacobianType](#)

Definition at line 764 of file types.f90.

**7.247.2.8 INTEGER(INTG) TYPES::FIELD\_INTERPOLATED\_POINT\_METRICS\_-  
TYPE::NUMBER\_OF\_X\_DIMENSIONS**

The number of X dimensions.

Definition at line 757 of file types.f90.

**7.247.2.9 INTEGER(INTG) TYPES::FIELD\_INTERPOLATED\_POINT\_METRICS\_-  
TYPE::NUMBER\_OF\_XI\_DIMENSIONS**

The number of Xi dimensions.

Definition at line 758 of file types.f90.

## 7.248 TYPES::FIELD\_INTERPOLATED\_POINT\_TYPE Struct Reference

Contains the interpolated value (and the derivatives wrt xi) of a field at a point. Old [CMISS](#) name XG.

Collaboration diagram for TYPES::FIELD\_INTERPOLATED\_POINT\_TYPE:

### Public Attributes

- TYPE(FIELD\_INTERPOLATION\_PARAMETERS\_TYPE), pointer INTERPOLATION\_PARAMETERS  
*A pointer to the interpolation parameters of the field that is to be interpolated.*
- INTEGER(INTG) MAX\_PARTIAL\_DERIVATIVE\_INDEX  
*The maximum number of partial derivatives that have been allocated for the values component.*
- INTEGER(INTG) PARTIAL\_DERIVATIVE\_TYPE  
*The type of the partial derivatives that have been interpolated. PARTIAL\_DERIVATIVE\_TYPE can be either NO\_PART\_DERIV, FIRST\_PART\_DERIV or SECOND\_PART\_DERIV depending on whether just the field value, the field value and all first derivatives (including cross derivatives) or the first value and all first and second derivatives have been interpolated.*
- REAL(DP), allocatable VALUES  
*VALUES(component\_idx,nu). The interpolated field components and their partial derivatives.*

### 7.248.1 Detailed Description

Contains the interpolated value (and the derivatives wrt xi) of a field at a point. Old [CMISS](#) name XG.

Definition at line 768 of file types.f90.

### 7.248.2 Member Data Documentation

#### 7.248.2.1 TYPE(FIELD\_INTERPOLATION\_PARAMETERS\_TYPE),pointer TYPES::FIELD\_INTERPOLATED\_POINT\_TYPE::INTERPOLATION\_PARAMETERS

A pointer to the interpolation parameters of the field that is to be interpolated.

Definition at line 769 of file types.f90.

#### 7.248.2.2 INTEGER(INTG) TYPES::FIELD\_INTERPOLATED\_POINT\_TYPE::MAX\_PARTIAL\_DERIVATIVE\_INDEX

The maximum number of partial derivatives that have been allocated for the values component.

Definition at line 770 of file types.f90.

**7.248.2.3 INTEGER(INTG) TYPES::FIELD\_INTERPOLATED\_POINT\_TYPE::PARTIAL\_DERIVATIVE\_TYPE**

The type of the partial derivatives that have been interpolated. PARTIAL\_DERIVATIVE\_TYPE can be either NO\_PART\_DERIV, FIRST\_PART\_DERIV or SECOND\_PART\_DERIV depending on whether just the field value, the field value and all first derivatives (including cross derivatives) or the first value and all first and second derivatives have been interpolated.

Definition at line 771 of file types.f90.

**7.248.2.4 REAL(DP),allocatable TYPES::FIELD\_INTERPOLATED\_POINT\_TYPE::VALUES**

VALUES(component\_idx,nu). The interpolated field components and their partial derivatives.

Definition at line 772 of file types.f90.

## 7.249 TYPES::FIELD\_INTERPOLATION\_PARAMETERS\_- TYPE Struct Reference

Contains the parameters required to interpolate a field variable within an element. Old [CMISS](#) name XE.

Collaboration diagram for TYPES::FIELD\_INTERPOLATION\_PARAMETERS\_TYPE:

### Public Attributes

- TYPE([FIELD\\_TYPE](#)), pointer **FIELD**  
*A pointer to the field to be interpolated.*
- TYPE([FIELD\\_VARIABLE\\_TYPE](#)), pointer **FIELD\_VARIABLE**  
*A pointer to the field VARIABLE to be interpolated.*
- TYPE([BASIS\\_PTR\\_TYPE](#)), allocatable **BASES**  
*BASES(component\_idx). An array to hold a pointer to the basis (if any) used for interpolating the component\_idx'th component of the field variable.*
- INTEGER([INTG](#)), allocatable **NUMBER\_OF\_PARAMETERS**  
*NUMBER\_OF\_PARAMETERS(component\_idx). The number of interpolation parameters used for interpolating the component\_idx'th component of the field variable.*
- REAL([DP](#)), allocatable **PARAMETERS**  
*PARAMETERS(ns,component\_idx). The interpolation parameters used for interpolating the component\_idx'th component of the field variable.*

### 7.249.1 Detailed Description

Contains the parameters required to interpolate a field variable within an element. Old [CMISS](#) name XE.

Definition at line 776 of file types.f90.

### 7.249.2 Member Data Documentation

#### 7.249.2.1 TYPE([BASIS\\_PTR\\_TYPE](#)),allocatable TYPES::FIELD\_INTERPOLATION\_- PARAMETERS\_TYPE::BASES

BASES(component\_idx). An array to hold a pointer to the basis (if any) used for interpolating the component\_idx'th component of the field variable.

Definition at line 779 of file types.f90.

#### 7.249.2.2 TYPE([FIELD\\_TYPE](#)),pointer TYPES::FIELD\_INTERPOLATION\_PARAMETERS\_- TYPE::FIELD

A pointer to the field to be interpolated.

Definition at line 777 of file types.f90.

**7.249.2.3 TYPE(FIELD\_VARIABLE\_TYPE),pointer TYPES::FIELD\_INTERPOLATION\_-  
PARAMETERS\_TYPE::FIELD\_VARIABLE**

A pointer to the field VARIABLE to be interpolated.

Definition at line 778 of file types.f90.

**7.249.2.4 INTEGER(INTG),allocatable TYPES::FIELD\_INTERPOLATION\_PARAMETERS\_-  
TYPE::NUMBER\_OF\_PARAMETERS**

NUMBER\_OF\_PARAMETERS(component\_idx). The number of interpolation parameters used for interpolating the component\_idx'th component of the field variable.

Definition at line 780 of file types.f90.

**7.249.2.5 REAL(DP),allocatable TYPES::FIELD\_INTERPOLATION\_PARAMETERS\_-  
TYPE::PARAMETERS**

PARAMETERS(ns,component\_idx). The interpolation parameters used for interpolating the component\_idx'th component of the field variable.

Definition at line 781 of file types.f90.

## 7.250 TYPES::FIELD\_MAPPINGS\_TYPE Struct Reference

The type containing the mappings for the field.

Collaboration diagram for TYPES::FIELD\_MAPPINGS\_TYPE:

### Public Attributes

- TYPE(FIELD\_DOF\_TO\_PARAM\_MAP\_TYPE) DOF\_TO\_PARAM\_MAP  
*The mappings for the field dofs to the field parameters.*
- TYPE(DOMAIN\_MAPPING\_TYPE), pointer DOMAIN\_MAPPING  
*The domain mappings for the field dofs i.e., the local to global mpapings etc.*

### 7.250.1 Detailed Description

The type containing the mappings for the field.

Definition at line 885 of file types.f90.

### 7.250.2 Member Data Documentation

#### 7.250.2.1 TYPE(FIELD\_DOF\_TO\_PARAM\_MAP\_TYPE) TYPES::FIELD\_MAPPINGS\_- TYPE::DOF\_TO\_PARAM\_MAP

The mappings for the field dofs to the field parameters.

Definition at line 886 of file types.f90.

#### 7.250.2.2 TYPE(DOMAIN\_MAPPING\_TYPE),pointer TYPES::FIELD\_MAPPINGS\_- TYPE::DOMAIN\_MAPPING

The domain mappings for the field dofs i.e., the local to global mpapings etc.

Definition at line 887 of file types.f90.

## 7.251 TYPES::FIELD\_PARAM\_TO\_DOF\_MAP\_TYPE Struct Reference

A type to hold the mapping from field parameters (nodes, elements, etc) to field dof numbers for a particular field variable component.

### Public Attributes

- INTEGER(INTG) [NUMBER\\_OF\\_CONSTANT\\_PARAMETERS](#)

*The number of constant field parameters for this field variable component. Note: this is currently always 1 but is.*

- INTEGER(INTG) [NUMBER\\_OF\\_ELEMENT\\_PARAMETERS](#)

*The number of element based field parameters for this field variable component.*

- INTEGER(INTG) [NUMBER\\_OF\\_NODE\\_PARAMETERS](#)

*The number of node based field parameters for this field variable component.*

- INTEGER(INTG) [MAX\\_NUMBER\\_OF\\_DERIVATIVES](#)

*The maximum number of derivatives for the node parameters for this field variable component. It is the size of the first index of the NODE\_PARAM2DOF\_MAP component of this type.*

- INTEGER(INTG) [NUMBER\\_OF\\_POINT\\_PARAMETERS](#)

*The number of point based field parameters for this field variable component.*

- INTEGER(INTG) [CONSTANT\\_PARAM2DOF\\_MAP](#)

*CONSTANT\_PARAM2DOF\_MAP(nc). The field dof (nc=0) or variable dof (nc=1) number of the constant parameter for this field variable component.*

- INTEGER(INTG), allocatable [ELEMENT\\_PARAM2DOF\\_MAP](#)

*ELEMENT\_PARAM2DOF\_MAP(ne,nc). The field dof (nc=0) or variable dof (nc=1) number of the ne'th element based parameter for this field variable component.*

- INTEGER(INTG), allocatable [NODE\\_PARAM2DOF\\_MAP](#)

*NODE\_PARAM2DOF\_MAP(nk,np,nc). The field dof (nc=0) or variable dof (nc=1) number of the nk'th derivative of the np'th node based parameter for this field variable component. Note: because the first index of this array is set to the maximum number of derivatives per node this array wastes memory if there are nodes with a smaller number of derivatives than the maximum.*

- INTEGER(INTG), allocatable [POINT\\_PARAM2DOF\\_MAP](#)

*POINT\_PARAM2DOF\_MAP(nq,nc). The field dof (nc=0) or variable dof (nc=1) number of nq'th point based parameter for this field variable component.*

### 7.251.1 Detailed Description

A type to hold the mapping from field parameters (nodes, elements, etc) to field dof numbers for a particular field variable component.

Definition at line 827 of file types.f90.

## 7.251.2 Member Data Documentation

### 7.251.2.1 INTEGER(INTG) TYPES::FIELD\_PARAM\_TO\_DOF\_MAP\_TYPE::CONSTANT\_PARAM2DOF\_MAP

CONSTANT\_PARAM2DOF\_MAP(nc). The field dof (nc=0) or variable dof (nc=1) number of the constant parameter for this field variable component.

Definition at line 834 of file types.f90.

### 7.251.2.2 INTEGER(INTG),allocatable TYPES::FIELD\_PARAM\_TO\_DOF\_MAP\_TYPE::ELEMENT\_PARAM2DOF\_MAP

ELEMENT\_PARAM2DOF\_MAP(ne,nc). The field dof (nc=0) or variable dof (nc=1) number of the ne'th element based parameter for this field variable component.

#### **Todo**

Allow for multiple element parameters per element.

Definition at line 835 of file types.f90.

### 7.251.2.3 INTEGER(INTG) TYPES::FIELD\_PARAM\_TO\_DOF\_MAP\_TYPE::MAX\_NUMBER\_OF\_DERIVATIVES

The maximum number of derivatives for the node parameters for this field variable component. It is the size of the first index of the NODE\_PARAM2DOF\_MAP component of this type.

Definition at line 832 of file types.f90.

### 7.251.2.4 INTEGER(INTG),allocatable TYPES::FIELD\_PARAM\_TO\_DOF\_MAP\_TYPE::NODE\_PARAM2DOF\_MAP

NODE\_PARAM2DOF\_MAP(nk,np,nc). The field dof (nc=0) or variable dof (nc=1) number of the nk'th derivative of the np'th node based parameter for this field variable component. Note: because the first index of this array is set to the maximum number of derivatives per node this array wastes memory if there are nodes with a smaller number of derivatives than the maximum.

#### **Todo**

Don't allocate too much memory if there are different numbers of derivatives for different nodes.

Definition at line 836 of file types.f90.

### 7.251.2.5 INTEGER(INTG) TYPES::FIELD\_PARAM\_TO\_DOF\_MAP\_TYPE::NUMBER\_OF\_CONSTANT\_PARAMETERS

The number of constant field parameters for this field variable component. Note: this is currently always 1 but is.

Definition at line 828 of file types.f90.

**7.251.2.6 INTEGER(INTG) TYPES::FIELD\_PARAM\_TO\_DOF\_MAP\_TYPE::NUMBER\_OF\_-ELEMENT\_PARAMETERS**

The number of element based field parameters for this field variable component.

Definition at line 830 of file types.f90.

**7.251.2.7 INTEGER(INTG) TYPES::FIELD\_PARAM\_TO\_DOF\_MAP\_TYPE::NUMBER\_OF\_-NODE\_PARAMETERS**

The number of node based field parameters for this field variable component.

Definition at line 831 of file types.f90.

**7.251.2.8 INTEGER(INTG) TYPES::FIELD\_PARAM\_TO\_DOF\_MAP\_TYPE::NUMBER\_OF\_-POINT\_PARAMETERS**

The number of point based field parameters for this field variable component.

Definition at line 833 of file types.f90.

**7.251.2.9 INTEGER(INTG),allocatable TYPES::FIELD\_PARAM\_TO\_DOF\_MAP\_-TYPE::POINT\_PARAM2DOF\_MAP**

POINT\_PARAM2DOF\_MAP(nq,nc). The field dof (nc=0) or variable dof (nc=1) number of nq'th point based parameter for this field variable component.

Definition at line 837 of file types.f90.

## 7.252 TYPES::FIELD\_PARAMETER\_SET\_PTR\_TYPE Struct Reference

A buffer type to allow for an array of pointers to a FIELD\_PARAMETER\_SET\_TYPE.

Collaboration diagram for TYPES::FIELD\_PARAMETER\_SET\_PTR\_TYPE:

### Public Attributes

- TYPE(FIELD\_PARAMETER\_SET\_TYPE), pointer PTR  
*The pointer to the field parameter set.*

#### 7.252.1 Detailed Description

A buffer type to allow for an array of pointers to a FIELD\_PARAMETER\_SET\_TYPE.

Definition at line 899 of file types.f90.

#### 7.252.2 Member Data Documentation

##### 7.252.2.1 TYPE(FIELD\_PARAMETER\_SET\_TYPE),pointer TYPES::FIELD\_PARAMETER\_SET\_PTR\_TYPE::PTR

The pointer to the field parameter set.

Definition at line 900 of file types.f90.

## 7.253 TYPES::FIELD\_PARAMETER\_SET\_TYPE Struct Reference

A type to hold the parameter sets for a field.

Collaboration diagram for TYPES::FIELD\_PARAMETER\_SET\_TYPE:

### Public Attributes

- INTEGER(INTG) [SET\\_INDEX](#)

*The global set index (from 1 to the [TYPES::FIELD\\_PARAMETER\\_SETS\\_TYPE::NUMBER\\_OF\\_PARAMETER\\_SETS](#)) that this parameter set corresponds to.*

- INTEGER(INTG) [SET\\_TYPE](#)

*The user set type (index) (from 1 to [FIELD\\_ROUTINES::FIELD\\_NUMBER\\_OF\\_SET\\_TYPES](#)) that this parameter set.*

- TYPE([DISTRIBUTED\\_VECTOR\\_TYPE](#)), pointer [PARAMETERS](#)

*A pointer to the distributed vector that contains the field parameters for this field parameter set.*

### 7.253.1 Detailed Description

A type to hold the parameter sets for a field.

Definition at line 891 of file types.f90.

### 7.253.2 Member Data Documentation

#### 7.253.2.1 TYPE([DISTRIBUTED\\_VECTOR\\_TYPE](#)),pointer [TYPES::FIELD\\_PARAMETER\\_SET\\_TYPE::PARAMETERS](#)

A pointer to the distributed vector that contains the field parameters for this field parameter set.

Definition at line 895 of file types.f90.

#### 7.253.2.2 INTEGER(INTG) [TYPES::FIELD\\_PARAMETER\\_SET\\_TYPE::SET\\_INDEX](#)

The global set index (from 1 to the [TYPES::FIELD\\_PARAMETER\\_SETS\\_TYPE::NUMBER\\_OF\\_PARAMETER\\_SETS](#)) that this parameter set corresponds to.

Definition at line 892 of file types.f90.

#### 7.253.2.3 INTEGER(INTG) [TYPES::FIELD\\_PARAMETER\\_SET\\_TYPE::SET\\_TYPE](#)

The user set type (index) (from 1 to [FIELD\\_ROUTINES::FIELD\\_NUMBER\\_OF\\_SET\\_TYPES](#)) that this parameter set.

See also:

[FIELD\\_ROUTINES::ParameterSetTypes](#)

Definition at line 893 of file types.f90.

## 7.254 TYPES::FIELD\_PARAMETER\_SETS\_TYPE Struct Reference

A type to store the parameter sets for a field.

Collaboration diagram for TYPES::FIELD\_PARAMETER\_SETS\_TYPE:

### Public Attributes

- INTEGER(INTG) [NUMBER\\_OF\\_PARAMETER\\_SETS](#)

*The number of parameter sets that are currently defined on the field.*

- TYPE([FIELD\\_TYPE](#)), pointer [FIELD](#)

*A pointer to the field that these parameter sets are defined on.*

- TYPE([FIELD\\_PARAMETER\\_SET\\_PTR\\_TYPE](#)), pointer [SET\\_TYPE](#)

*SET\_TYPE(set\_type\_idx). A pointer to an array of pointers to the field set types. SET\_TYPE(set\_type\_idx)PTR is a pointer to the parameter set type for the set\_type\_idx'th parameter set. set\_type\_idx can vary from 1 to [FIELD\\_ROUTINES::FIELD\\_NUMBER\\_OF\\_SET\\_TYPES](#). The value of the pointer will be NULL if the parameter set corresponding to the set\_type\_idx'th parameter set has not yet been created for the field.*

- TYPE([FIELD\\_PARAMETER\\_SET\\_PTR\\_TYPE](#)), pointer [PARAMETER\\_SETS](#)

*PARAMETER\_SETS(set\_type\_idx). A pointer to an array of pointers to the parameter sets that have been created on the field. PARAMETER\_SET(set\_type\_idx)PTR is a pointer to the parameter set type for the set\_type\_idx'th parameter set that has been created. set\_type\_idx can vary from 1 to the number of parameter set types that have currently been created for the field i.e., [TYPES::FIELD\\_PARAMETER\\_SETS\\_TYPE::NUMBER\\_OF\\_PARAMETER\\_SETS](#).*

### 7.254.1 Detailed Description

A type to store the parameter sets for a field.

Definition at line 904 of file types.f90.

### 7.254.2 Member Data Documentation

#### 7.254.2.1 TYPE([FIELD\\_TYPE](#)),pointer [TYPES::FIELD\\_PARAMETER\\_SETS\\_TYPE::FIELD](#)

A pointer to the field that these parameter sets are defined on.

Definition at line 906 of file types.f90.

#### 7.254.2.2 INTEGER(INTG) [TYPES::FIELD\\_PARAMETER\\_SETS\\_TYPE::NUMBER\\_OF\\_PARAMETER\\_SETS](#)

The number of parameter sets that are currently defined on the field.

Definition at line 905 of file types.f90.

**7.254.2.3 TYPE(FIELD\_PARAMETER\_SET\_PTR\_TYPE),pointer  
TYPES::FIELD\_PARAMETER\_SETS\_TYPE::PARAMETER\_SETS**

PARAMETER\_SETS(set\_type\_idx). A pointer to an array of pointers to the parameter sets that have been created on the field. PARAMETER\_SET(set\_type\_idx)PTR is a pointer to the parameter set type for the set\_type\_idx'th parameter set that has been created. set\_type\_idx can vary from 1 to the number of parameter set types that have currently been created for the field i.e., [TYPES::FIELD\\_PARAMETER\\_SETS\\_TYPE::NUMBER\\_OF\\_PARAMETER\\_SETS](#).

Definition at line 908 of file types.f90.

**7.254.2.4 TYPE(FIELD\_PARAMETER\_SET\_PTR\_TYPE),pointer  
TYPES::FIELD\_PARAMETER\_SETS\_TYPE::SET\_TYPE**

SET\_TYPE(set\_type\_idx). A pointer to an array of pointers to the field set types. SET\_TYPE(set\_type\_idx)PTR is a pointer to the parameter set type for the set\_type\_idx'th parameter set. set\_type\_idx can vary from 1 to [FIELD\\_ROUTINES::FIELD\\_NUMBER\\_OF\\_SET\\_TYPES](#). The value of the pointer will be NULL if the parameter set corresponding to the set\_type\_idx'th parameter set has not yet been created for the field.

Definition at line 907 of file types.f90.

## 7.255 TYPES::FIELD\_PTR\_TYPE Struct Reference

A buffer type to allow for an array of pointers to a [FIELD\\_TYPE](#).

Collaboration diagram for TYPES::FIELD\_PTR\_TYPE:

### Public Attributes

- TYPE([FIELD\\_TYPE](#)), pointer PTR

*The pointer to the field.*

### 7.255.1 Detailed Description

A buffer type to allow for an array of pointers to a [FIELD\\_TYPE](#).

Definition at line 934 of file types.f90.

### 7.255.2 Member Data Documentation

#### 7.255.2.1 TYPE(FIELD\_TYPE),pointer TYPES::FIELD\_PTR\_TYPE::PTR

The pointer to the field.

Definition at line 935 of file types.f90.

## 7.256 TYPES::FIELD\_SCALING\_TYPE Struct Reference

A type to hold the scale factors for the appropriate mesh component of a field.

Collaboration diagram for TYPES::FIELD\_SCALING\_TYPE:

### Public Attributes

- INTEGER(INTG) [MESH\\_COMPONENT\\_NUMBER](#)

*The mesh component number of a field variable component that the scaling factors are associated with.*

- INTEGER(INTG) [MAX\\_NUMBER\\_OF\\_DERIVATIVES](#)

*The maximum number of derivatives in the mesh component.*

- INTEGER(INTG) [MAX\\_NUMBER\\_OF\\_ELEMENT\\_PARAMETERS](#)

*The maximum number of element parameters in the mesh component.*

- TYPE([DISTRIBUTED\\_VECTOR\\_TYPE](#)), pointer [SCALE\\_FACTORS](#)

*SCALE\_FACTORS( $nk, np$ ). The scale factor that is applied to the  $nk$ 'th derivative of the  $np$ 'th node of the mesh component.*

### 7.256.1 Detailed Description

A type to hold the scale factors for the appropriate mesh component of a field.

Definition at line 797 of file types.f90.

### 7.256.2 Member Data Documentation

#### 7.256.2.1 INTEGER(INTG) TYPES::FIELD\_SCALING\_TYPE::MAX\_NUMBER\_OF\_DERIVATIVES

The maximum number of derivatives in the mesh component.

Definition at line 799 of file types.f90.

#### 7.256.2.2 INTEGER(INTG) TYPES::FIELD\_SCALING\_TYPE::MAX\_NUMBER\_OF\_ELEMENT\_PARAMETERS

The maximum number of element parameters in the mesh component.

Definition at line 800 of file types.f90.

#### 7.256.2.3 INTEGER(INTG) TYPES::FIELD\_SCALING\_TYPE::MESH\_COMPONENT\_NUMBER

The mesh component number of a field variable component that the scaling factors are associated with.

Definition at line 798 of file types.f90.

**7.256.2.4 TYPE(DISTRIBUTED\_VECTOR\_TYPE),pointer TYPES::FIELD\_SCALING\_-  
TYPE::SCALE\_FACTORS**

SCALE\_FACTORS(nk,np). The scale factor that is applied to the nk'th derivative of the np'th node of the mesh component.

**Todo**

Make scale factors nodally based for now. Will have to revert to element based and extended to be a matrix to allow for a global derivative to be mapped onto many different element derivatives at the points that closes meshes or for inconsistent xi directions

Definition at line 801 of file types.f90.

## 7.257 TYPES::FIELD\_SCALINGS\_TYPE Struct Reference

A type to hold the field scalings for the field.

Collaboration diagram for TYPES::FIELD\_SCALINGS\_TYPE:

### Public Attributes

- INTEGER(INTG) [SCALING\\_TYPE](#)

*The type of scaling that is applied to the field.*

- INTEGER(INTG) [NUMBER\\_OF\\_SCALING\\_INDICES](#)

*The number of scaling indices (or sets of scale factors) for the field. In general there will be a set of scale factors (or a scaling index) for each different mesh component that is used by the field variable components.*

- TYPE([FIELD\\_SCALING\\_TYPE](#)), allocatable [SCALINGS](#)

*SCALINGS(scaling\_idx). The scaling factors for the scaling\_idx'th set of scaling factors.*

### 7.257.1 Detailed Description

A type to hold the field scalings for the field.

Definition at line 805 of file types.f90.

### 7.257.2 Member Data Documentation

#### 7.257.2.1 INTEGER(INTG) TYPES::FIELD\_SCALINGS\_TYPE::NUMBER\_OF\_SCALING\_INDICES

The number of scaling indices (or sets of scale factors) for the field. In general there will be a set of scale factors (or a scaling index) for each different mesh component that is used by the field variable components.

Definition at line 807 of file types.f90.

#### 7.257.2.2 INTEGER(INTG) TYPES::FIELD\_SCALINGS\_TYPE::SCALING\_TYPE

The type of scaling that is applied to the field.

See also:

[FIELD\\_ROUTINES::ScalingTypes](#)

Definition at line 806 of file types.f90.

#### 7.257.2.3 TYPE(FIELD\_SCALING\_TYPE),allocatable TYPES::FIELD\_SCALINGS\_TYPE::SCALINGS

SCALINGS(scaling\_idx). The scaling factors for the scaling\_idx'th set of scaling factors.

Definition at line 808 of file types.f90.

## 7.258 TYPES::FIELD\_TYPE Struct Reference

Contains information for a field defined on a region.

Collaboration diagram for TYPES::FIELD\_TYPE:

### Public Attributes

- INTEGER(INTG) [GLOBAL\\_NUMBER](#)

*The global number of the field in the list of fields for a region.*

- INTEGER(INTG) [USER\\_NUMBER](#)

*The user defined identifier for the field. The user number must be unique.*

- LOGICAL [FIELD\\_FINISHED](#)

*Is .TRUE. if the field has finished being created, .FALSE. if not.*

- TYPE(FIELDS\_TYPE), pointer [FIELDS](#)

*A pointer to the fields for this region.*

- TYPE(REGION\_TYPE), pointer [REGION](#)

*A pointer to the region for this field.*

- INTEGER(INTG) [TYPE](#)

*The type of the field.*

- INTEGER(INTG) [DEPENDENT\\_TYPE](#)

*The dependent type of the field.*

- INTEGER(INTG) [DIMENSION](#)

*The dimension of the field.*

- TYPE(DECOMPOSITION\_TYPE), pointer [DECOMPOSITION](#)

*A pointer to the decomposition of the mesh for which the field is defined on.*

- INTEGER(INTG) [NUMBER\\_OF\\_VARIABLES](#)

*The number of variable types in the field. Old CMISS name NCT(nr;nx).*

- TYPE(FIELD\_VARIABLE\_PTR\_TYPE), allocatable [VARIABLE\\_TYPE\\_MAP](#)

*VARIABLE\_TYPE\_MAP(variable\_idx). The map from the available field variable types to the field variable types that are defined for the field. variable\_idx varies from 1 to [FIELD\\_ROUTINES::FIELD\\_NUMBER\\_OF\\_VARIABLE\\_TYPES](#). If the particular field variable type has not been defined on the field then the VARIABLE\_TYPE\_MAP will be NULL.*

- TYPE(FIELD\_VARIABLE\_TYPE), allocatable [VARIABLES](#)

*VARIABLES(variable\_idx). The array of field variables.*

- TYPE(FIELD\_SCALINGS\_TYPE) [SCALINGS](#)

*The scaling parameters for the field.*

- **TYPE(FIELD\_MAPPINGS\_TYPE) MAPPINGS**

*The mappings for the field.*

- **TYPE(FIELD\_PARAMETER\_SETS\_TYPE) PARAMETER\_SETS**

*The parameter sets for the field.*

- **TYPE(FIELD\_TYPE), pointer GEOMETRIC\_FIELD**

*A pointer to the geometric field that this field uses. If the field itself is a geometric field then this will be a pointer back to itself.*

- **TYPE(FIELD\_GEOMETRIC\_PARAMETERS\_TYPE), pointer GEOMETRIC\_FIELD\_PARAMETERS**

*TYPE(FIELD\_CREATE\_VALUES\_CACHE\_TYPE), POINTER :: CREATE\_VALUES\_CACHE !<The create values cache for the field.*

### 7.258.1 Detailed Description

Contains information for a field defined on a region.

Definition at line 912 of file types.f90.

### 7.258.2 Member Data Documentation

#### 7.258.2.1 TYPE(DECOMPOSITION\_TYPE),pointer TYPES::FIELD\_TYPE::DECOMPOSITION

A pointer to the decomposition of the mesh for which the field is defined on.

Definition at line 921 of file types.f90.

#### 7.258.2.2 INTEGER(INTG) TYPES::FIELD\_TYPE::DEPENDENT\_TYPE

The dependent type of the field.

**See also:**

[FIELD\\_ROUTINES::DependentTypes](#)

Definition at line 919 of file types.f90.

#### 7.258.2.3 INTEGER(INTG) TYPES::FIELD\_TYPE::DIMENSION

The dimension of the field.

**See also:**

[FIELD\\_ROUTINES::DimensionTypes](#)

Definition at line 920 of file types.f90.

**7.258.2.4 LOGICAL TYPES::FIELD\_TYPE::FIELD\_FINISHED**

Is .TRUE. if the field has finished being created, .FALSE. if not.

Definition at line 915 of file types.f90.

**7.258.2.5 TYPE(FIELDS\_TYPE),pointer TYPES::FIELD\_TYPE::FIELDS**

A pointer to the fields for this region.

Definition at line 916 of file types.f90.

**7.258.2.6 TYPE(FIELD\_TYPE),pointer TYPES::FIELD\_TYPE::GEOMETRIC\_FIELD**

A pointer to the geometric field that this field uses. If the field itself is a geometric field then this will be a pointer back to itself.

Definition at line 928 of file types.f90.

**7.258.2.7 TYPE(FIELD\_GEOMETRIC\_PARAMETERS\_TYPE),pointer  
TYPES::FIELD\_TYPE::GEOMETRIC\_FIELD\_PARAMETERS**

TYPE(FIELD\_CREATE\_VALUES\_CACHE\_TYPE), POINTER :: CREATE\_VALUES\_CACHE !<The create values cache for the field.

Definition at line 929 of file types.f90.

**7.258.2.8 INTEGER(INTG) TYPES::FIELD\_TYPE::GLOBAL\_NUMBER**

The global number of the field in the list of fields for a region.

Definition at line 913 of file types.f90.

**7.258.2.9 TYPE(FIELD\_MAPPINGS\_TYPE) TYPES::FIELD\_TYPE::MAPPINGS**

The mappings for the field.

Definition at line 926 of file types.f90.

**7.258.2.10 INTEGER(INTG) TYPES::FIELD\_TYPE::NUMBER\_OF\_VARIABLES**

The number of variable types in the field. Old CMIS name NCT(nr,nx).

Definition at line 922 of file types.f90.

**7.258.2.11 TYPE(FIELD\_PARAMETER\_SETS\_TYPE) TYPES::FIELD\_-  
TYPE::PARAMETER\_SETS**

The parameter sets for the field.

Definition at line 927 of file types.f90.

**7.258.2.12 TYPE(REGION\_TYPE),pointer TYPES::FIELD\_TYPE::REGION**

A pointer to the region for this field.

Definition at line 917 of file types.f90.

**7.258.2.13 TYPE(FIELD\_SCALINGS\_TYPE) TYPES::FIELD\_TYPE::SCALINGS**

The scaling parameters for the field.

Definition at line 925 of file types.f90.

**7.258.2.14 INTEGER(INTG) TYPES::FIELD\_TYPE::TYPE**

The type of the field.

**See also:**

[FIELD\\_ROUTINES::FieldTypes](#)

Definition at line 918 of file types.f90.

**7.258.2.15 INTEGER(INTG) TYPES::FIELD\_TYPE::USER\_NUMBER**

The user defined identifier for the field. The user number must be unique.

Definition at line 914 of file types.f90.

**7.258.2.16 TYPE(FIELD\_VARIABLE\_PTR\_TYPE),allocatable TYPES::FIELD\_TYPE::VARIABLE\_TYPE\_MAP**

VARIABLE\_TYPE\_MAP(variable\_idx). The map from the available field variable types to the field variable types that are defined for the field. variable\_idx varies from 1 to [FIELD\\_ROUTINES::FIELD\\_NUMBER\\_OF\\_VARIABLE\\_TYPES](#). If the particular field variable type has not been defined on the field then the VARIABLE\_TYPE\_MAP will be NULL.

**See also:**

[FIELD\\_ROUTINES::VariableTypes](#)

Definition at line 923 of file types.f90.

**7.258.2.17 TYPE(FIELD\_VARIABLE\_TYPE),allocatable TYPES::FIELD\_TYPE::VARIABLES**

VARIABLES(variable\_idx) .The array of field variables.

Definition at line 924 of file types.f90.

## 7.259 TYPES::FIELD\_VARIABLE\_COMPONENT\_TYPE Struct Reference

Contains information for a component of a field variable.

Collaboration diagram for TYPES::FIELD\_VARIABLE\_COMPONENT\_TYPE:

### Public Attributes

- INTEGER(INTG) **COMPONENT\_NUMBER**  
*The number of the field variable component.*
- TYPE(FIELD\_VARIABLE\_TYPE), pointer **FIELD\_VARIABLE**  
*A pointer to the field variable for this component.*
- TYPE(FIELD\_TYPE), pointer **FIELD**  
*A pointer to the field for this field variable component.*
- TYPE(REGION\_TYPE), pointer **REGION**  
*A pointer to the region for this field variable component.*
- INTEGER(INTG) **INTERPOLATION\_TYPE**  
*The interpolation type of the field variable component.*
- INTEGER(INTG) **MESH\_COMPONENT\_NUMBER**  
*The mesh component of the field decomposition for this field variable component.*
- INTEGER(INTG) **SCALING\_INDEX**  
*The index into the defined field scalings for this field variable component.*
- TYPE(DOMAIN\_TYPE), pointer **DOMAIN**  
*A pointer to the domain of the field decomposition for this field variable component.*
- INTEGER(INTG) **MAX\_NUMBER\_OF\_INTERPOLATION\_PARAMETERS**  
*The maximum number of interpolations parameters in an element for a field variable component.*
- TYPE(FIELD\_PARAM\_TO\_DOF\_MAP\_TYPE) **PARAM\_TO\_DOF\_MAP**  
*The mapping of the field parameters to the field dofs for this field variable component.*

### 7.259.1 Detailed Description

Contains information for a component of a field variable.

Definition at line 841 of file types.f90.

## 7.259.2 Member Data Documentation

### 7.259.2.1 INTEGER(INTG) TYPES::FIELD\_VARIABLE\_COMPONENT\_- TYPE::COMPONENT\_NUMBER

The number of the field variable component.

Definition at line 842 of file types.f90.

### 7.259.2.2 TYPE(DOMAIN\_TYPE),pointer TYPES::FIELD\_VARIABLE\_COMPONENT\_- TYPE::DOMAIN

A pointer to the domain of the field decomposition for this field variable component.

Definition at line 849 of file types.f90.

### 7.259.2.3 TYPE(FIELD\_TYPE),pointer TYPES::FIELD\_VARIABLE\_COMPONENT\_- TYPE::FIELD

A pointer to the field for this field variable component.

Definition at line 844 of file types.f90.

### 7.259.2.4 TYPE(FIELD\_VARIABLE\_TYPE),pointer TYPES::FIELD\_VARIABLE\_- COMPONENT\_TYPE::FIELD\_VARIABLE

A pointer to the field variable for this component.

Definition at line 843 of file types.f90.

### 7.259.2.5 INTEGER(INTG) TYPES::FIELD\_VARIABLE\_COMPONENT\_- TYPE::INTERPOLATION\_TYPE

The interpolation type of the field variable component.

See also:

[FIELD\\_ROUTINES::InterpolationTypes](#)

Definition at line 846 of file types.f90.

### 7.259.2.6 INTEGER(INTG) TYPES::FIELD\_VARIABLE\_COMPONENT\_TYPE::MAX\_- NUMBER\_OF\_INTERPOLATION\_PARAMETERS

The maximum number of interpolations parameters in an element for a field variable component.

Definition at line 850 of file types.f90.

### 7.259.2.7 INTEGER(INTG) TYPES::FIELD\_VARIABLE\_COMPONENT\_TYPE::MESH\_- COMPONENT\_NUMBER

The mesh component of the field decomposition for this field variable component.

Definition at line 847 of file types.f90.

**7.259.2.8 TYPE(FIELD\_PARAM\_TO\_DOF\_MAP\_TYPE) TYPES::FIELD\_VARIABLE\_COMPONENT\_TYPE::PARAM\_TO\_DOF\_MAP**

The mapping of the field parameters to the field dofs for this field variable component.

Definition at line 851 of file types.f90.

**7.259.2.9 TYPE(REGION\_TYPE),pointer TYPES::FIELD\_VARIABLE\_COMPONENT\_TYPE::REGION**

A pointer to the region for this field variable component.

Definition at line 845 of file types.f90.

**7.259.2.10 INTEGER(INTG) TYPES::FIELD\_VARIABLE\_COMPONENT\_TYPE::SCALING\_INDEX**

The index into the defined field scalings for this field variable component.

Definition at line 848 of file types.f90.

## 7.260 TYPES::FIELD\_VARIABLE\_PTR\_TYPE Struct Reference

A buffer type to allow for an array of pointers to a [FIELD\\_VARIABLE\\_TYPE](#).

Collaboration diagram for TYPES::FIELD\_VARIABLE\_PTR\_TYPE:

### Public Attributes

- TYPE([FIELD\\_VARIABLE\\_TYPE](#)), pointer PTR

*The pointer to the field variable.*

### 7.260.1 Detailed Description

A buffer type to allow for an array of pointers to a [FIELD\\_VARIABLE\\_TYPE](#).

Definition at line 872 of file types.f90.

### 7.260.2 Member Data Documentation

#### 7.260.2.1 TYPE(FIELD\_VARIABLE\_TYPE),pointer TYPES::FIELD\_VARIABLE\_PTR\_- TYPE::PTR

The pointer to the field variable.

Definition at line 873 of file types.f90.

## 7.261 TYPES::FIELD\_VARIABLE\_TYPE Struct Reference

Contains information for a field variable defined on a field.

Collaboration diagram for TYPES::FIELD\_VARIABLE\_TYPE:

### Public Attributes

- INTEGER(INTG) [VARIABLE\\_NUMBER](#)  
*The number of the field variable.*
- INTEGER(INTG) [VARIABLE\\_TYPE](#)  
*The type of the field variable.*
- TYPE([FIELD\\_TYPE](#)), pointer [FIELD](#)  
*A pointer to the field for this field variable.*
- TYPE([REGION\\_TYPE](#)), pointer [REGION](#)  
*A pointer to the region for this field variable.*
- INTEGER(INTG) [MAX\\_NUMBER\\_OF\\_INTERPOLATION\\_PARAMETERS](#)  
*The maximum number of interpolation parameters in an element for a field variable.*
- INTEGER(INTG) [NUMBER\\_OF\\_DOFS](#)  
*Number of local degrees of freedom for this field variable (excluding ghosted dofs). Old CMISS name NYNR(0,0,nc,nr,nx).*
- INTEGER(INTG) [TOTAL\\_NUMBER\\_OF\\_DOFS](#)  
*Number of local degrees of freedom for this field variable (including ghosted dofs). Old CMISS name NYNR(0,0,nc,nr,nx).*
- INTEGER(INTG) [NUMBER\\_OF\\_GLOBAL\\_DOFS](#)  
*Number of global degrees of freedom for this field variable. Old CMISS name NYNR(0,0,nc,nr,nx).*
- INTEGER(INTG) [GLOBAL\\_DOF\\_OFFSET](#)  
*The offset of the start of the global dofs for this variable in the list of global field dofs.*
- INTEGER(INTG), allocatable [DOF\\_LIST](#)  
*DOF\_LIST(i). The list of (local) field dofs in this field variable. Old CMISS name NYNR(1..,0,nc,nr,nx).*
- TYPE([DOMAIN\\_MAPPING\\_TYPE](#)), pointer [DOMAIN\\_MAPPING](#)  
*Domain mapping for this variable. Only allocated if the field is a dependent field. May have to reconsider this as we now have a domain mapping for the field as a whole and for each variable of the field. The variable domain mapping to is allow a global matrix/vector mapping to be obtained from the field variable.*
- INTEGER(INTG) [NUMBER\\_OF\\_COMPONENTS](#)  
*The number of components in the field variable.*
- TYPE([FIELD\\_VARIABLE\\_COMPONENT\\_TYPE](#)), allocatable [COMPONENTS](#)  
*COMPONENTS(component\_idx). The array of field variable components.*

### 7.261.1 Detailed Description

Contains information for a field variable defined on a field.

Definition at line 855 of file types.f90.

### 7.261.2 Member Data Documentation

#### 7.261.2.1 **TYPE(FIELD\_VARIABLE\_COMPONENT\_TYPE),allocatable TYPES::FIELD\_VARIABLE\_TYPE::COMPONENTS**

COMPONENTS(component\_idx). The array of field variable components.

Definition at line 868 of file types.f90.

#### 7.261.2.2 **INTEGER(INTG),allocatable TYPES::FIELD\_VARIABLE\_TYPE::DOF\_LIST**

DOF\_LIST(i). The list of (local) field dofs in this field variable. Old [CMISS](#) name NYNR(1..,0,nc,nr,nx).

Definition at line 865 of file types.f90.

#### 7.261.2.3 **TYPE(DOMAIN\_MAPPING\_TYPE),pointer TYPES::FIELD\_VARIABLE\_- TYPE::DOMAIN\_MAPPING**

Domain mapping for this variable. Only allocated if the field is a dependent field. May have to reconsider this as we now have a domain mapping for the field as a whole and for each variable of the field. The variable domain mapping to is allow a global matrix/vector mapping to be obtained from the field variable.

Definition at line 866 of file types.f90.

#### 7.261.2.4 **TYPE(FIELD\_TYPE),pointer TYPES::FIELD\_VARIABLE\_TYPE::FIELD**

A pointer to the field for this field variable.

Definition at line 858 of file types.f90.

#### 7.261.2.5 **INTEGER(INTG) TYPES::FIELD\_VARIABLE\_TYPE::GLOBAL\_DOF\_OFFSET**

The offset of the start of the global dofs for this variable in the list of global field dofs.

Definition at line 864 of file types.f90.

#### 7.261.2.6 **INTEGER(INTG) TYPES::FIELD\_VARIABLE\_TYPE::MAX\_NUMBER\_OF\_- INTERPOLATION\_PARAMETERS**

The maximum number of interpolation parameters in an element for a field variable.

Definition at line 860 of file types.f90.

**7.261.2.7 INTEGER(INTG) TYPES::FIELD\_VARIABLE\_TYPE::NUMBER\_OF\_COMPONENTS**

The number of components in the field variable.

Definition at line 867 of file types.f90.

**7.261.2.8 INTEGER(INTG) TYPES::FIELD\_VARIABLE\_TYPE::NUMBER\_OF\_DOFS**

Number of local degrees of freedom for this field variable (excluding ghosted dofs). Old **CMISS** name NYNR(0,0,nc,nr,nx).

Definition at line 861 of file types.f90.

**7.261.2.9 INTEGER(INTG) TYPES::FIELD\_VARIABLE\_TYPE::NUMBER\_OF\_GLOBAL\_DOFS**

Number of global degrees of freedom for this field variable. Old **CMISS** name NYNR(0,0,nc,nr,nx).

Definition at line 863 of file types.f90.

**7.261.2.10 TYPE(REGION\_TYPE),pointer TYPES::FIELD\_VARIABLE\_TYPE::REGION**

A pointer to the region for this field variable.

Definition at line 859 of file types.f90.

**7.261.2.11 INTEGER(INTG) TYPES::FIELD\_VARIABLE\_TYPE::TOTAL\_NUMBER\_OF\_DOFS**

Number of local degrees of freedom for this field variable (including ghosted dofs). Old **CMISS** name NYNR(0,0,nc,nr,nx).

Definition at line 862 of file types.f90.

**7.261.2.12 INTEGER(INTG) TYPES::FIELD\_VARIABLE\_TYPE::VARIABLE\_NUMBER**

The number of the field variable.

Definition at line 856 of file types.f90.

**7.261.2.13 INTEGER(INTG) TYPES::FIELD\_VARIABLE\_TYPE::VARIABLE\_TYPE**

The type of the field variable.

**See also:**

[FIELD\\_ROUTINES::VariableTypes](#)

Definition at line 857 of file types.f90.

## 7.262 TYPES::FIELDS\_TYPE Struct Reference

Contains information on the fields defined on a region.

Collaboration diagram for TYPES::FIELDS\_TYPE:

### Public Attributes

- TYPE([REGION\\_TYPE](#)), pointer **REGION**  
*A pointer to the region containing the fields.*
- INTEGER(INTG) **NUMBER\_OF\_FIELDS**  
*The number of fields defined on the region.*
- TYPE([FIELD\\_PTR\\_TYPE](#)), pointer **FIELDS**  
*FIELDS(fields\_idx). The array of pointers to the fields.*

### 7.262.1 Detailed Description

Contains information on the fields defined on a region.

Definition at line 939 of file types.f90.

### 7.262.2 Member Data Documentation

#### 7.262.2.1 TYPE(FIELD\_PTR\_TYPE),pointer TYPES::FIELDS\_TYPE::FIELDS

FIELDS(fields\_idx). The array of pointers to the fields.

Definition at line 942 of file types.f90.

#### 7.262.2.2 INTEGER(INTG) TYPES::FIELDS\_TYPE::NUMBER\_OF\_FIELDS

The number of fields defined on the region.

Definition at line 941 of file types.f90.

#### 7.262.2.3 TYPE(REGION\_TYPE),pointer TYPES::FIELDS\_TYPE::REGION

A pointer to the region containing the fields.

Definition at line 940 of file types.f90.

## 7.263 TYPES::GENERATED\_MESH\_PTR\_TYPE Struct Reference

A buffer type to allow for an array of pointers to a [GENERATED\\_MESH\\_TYPE](#).

Collaboration diagram for TYPES::GENERATED\_MESH\_PTR\_TYPE:

### Public Attributes

- TYPE([GENERATED\\_MESH\\_TYPE](#)), pointer PTR

*The pointer to the generated mesh.*

#### 7.263.1 Detailed Description

A buffer type to allow for an array of pointers to a [GENERATED\\_MESH\\_TYPE](#).

Definition at line 343 of file types.f90.

#### 7.263.2 Member Data Documentation

##### 7.263.2.1 TYPE(GENERATED\_MESH\_TYPE),pointer TYPES::GENERATED\_MESH\_PTR\_- TYPE::PTR

The pointer to the generated mesh.

Definition at line 344 of file types.f90.

## 7.264 TYPES::GENERATED\_MESH\_REGULAR\_TYPE Struct Reference

Contains information on a generated regular mesh.

Collaboration diagram for TYPES::GENERATED\_MESH\_REGULAR\_TYPE:

### Public Attributes

- TYPE(GENERATED\_MESH\_TYPE), pointer GENERATED\_MESH  
*A pointer to the generated mesh.*
- REAL(DP), allocatable ORIGIN  
*ORIGIN(nj). The position of the origin (first) corner of the regular mesh.*
- REAL(DP), allocatable MAXIMUM\_EXTENT  
*MAXIMUM\_EXTENT(nj). The extent/size in each nj'th direction of the regular mesh.*
- INTEGER(INTG) MESH\_DIMENSION  
*The dimension/number of Xi directions of the regular mesh.*
- INTEGER(INTG), allocatable NUMBER\_OF\_ELEMENTS\_XI  
*NUMBER\_OF\_ELEMENTS\_XI(ni). The number of elements in the ni'th Xi direction for the mesh.*
- TYPE(BASIS\_TYPE), pointer BASIS  
*The pointer to the basis used in the regular mesh.*

### 7.264.1 Detailed Description

Contains information on a generated regular mesh.

Definition at line 322 of file types.f90.

### 7.264.2 Member Data Documentation

#### 7.264.2.1 TYPE(BASIS\_TYPE),pointer TYPES::GENERATED\_MESH\_REGULAR\_- TYPE::BASIS

The pointer to the basis used in the regular mesh.

Definition at line 328 of file types.f90.

#### 7.264.2.2 TYPE(GENERATED\_MESH\_TYPE),pointer TYPES::GENERATED\_MESH\_- REGULAR\_TYPE::GENERATED\_MESH

A pointer to the generated mesh.

Definition at line 323 of file types.f90.

**7.264.2.3 REAL(DP),allocatable TYPES::GENERATED\_MESH\_REGULAR\_-  
TYPE::MAXIMUM\_EXTENT**

MAXIMUM\_EXTENT(nj). The extent/size in each nj'th direction of the regular mesh.

Definition at line 325 of file types.f90.

**7.264.2.4 INTEGER(INTG) TYPES::GENERATED\_MESH\_REGULAR\_TYPE::MESH\_-  
DIMENSION**

The dimension/number of Xi directions of the regular mesh.

Definition at line 326 of file types.f90.

**7.264.2.5 INTEGER(INTG),allocatable TYPES::GENERATED\_MESH\_REGULAR\_-  
TYPE::NUMBER\_OF\_ELEMENTS\_XI**

NUMBER\_OF\_ELEMENTS\_XI(ni). The number of elements in the ni'th Xi direction for the mesh.

Definition at line 327 of file types.f90.

**7.264.2.6 REAL(DP),allocatable TYPES::GENERATED\_MESH\_REGULAR\_TYPE::ORIGIN**

ORIGIN(nj). The position of the origin (first) corner of the regular mesh.

Definition at line 324 of file types.f90.

## 7.265 TYPES::GENERATED\_MESH\_TYPE Struct Reference

Contains information on a generated mesh.

Collaboration diagram for TYPES::GENERATED\_MESH\_TYPE:

### Public Attributes

- INTEGER(INTG) [USER\\_NUMBER](#)
- INTEGER(INTG) [GLOBAL\\_NUMBER](#)
- LOGICAL [GENERATED\\_MESH\\_FINISHED](#)

*Is .TRUE. if the generated mesh has finished being created, .FALSE. if not.*

- TYPE([REGION\\_TYPE](#)), pointer [REGION](#)
- INTEGER(INTG) [GENERATED\\_TYPE](#)
- TYPE([GENERATED\\_MESH\\_REGULAR\\_TYPE](#)), pointer [REGULAR\\_MESH](#)
- TYPE([MESH\\_TYPE](#)), pointer [MESH](#)

### 7.265.1 Detailed Description

Contains information on a generated mesh.

Definition at line 332 of file types.f90.

### 7.265.2 Member Data Documentation

#### 7.265.2.1 LOGICAL TYPES::GENERATED\_MESH\_TYPE::GENERATED\_MESH\_FINISHED

Is .TRUE. if the generated mesh has finished being created, .FALSE. if not.

Definition at line 335 of file types.f90.

#### 7.265.2.2 INTEGER(INTG) TYPES::GENERATED\_MESH\_TYPE::GENERATED\_TYPE

Definition at line 337 of file types.f90.

#### 7.265.2.3 INTEGER(INTG) TYPES::GENERATED\_MESH\_TYPE::GLOBAL\_NUMBER

Definition at line 334 of file types.f90.

#### 7.265.2.4 TYPE(MESH\_TYPE),pointer TYPES::GENERATED\_MESH\_TYPE::MESH

Definition at line 339 of file types.f90.

#### 7.265.2.5 TYPE(REGION\_TYPE),pointer TYPES::GENERATED\_MESH\_TYPE::REGION

Definition at line 336 of file types.f90.

**7.265.2.6 TYPE(GENERATED\_MESH\_REGULAR\_TYPE),pointer  
TYPES::GENERATED\_MESH\_TYPE::REGULAR\_MESH**

Definition at line 338 of file types.f90.

**7.265.2.7 INTEGER(INTG) TYPES::GENERATED\_MESH\_TYPE::USER\_NUMBER**

Definition at line 333 of file types.f90.

## 7.266 TYPES::GENERATED\_MESHES\_TYPE Struct Reference

Contains information on the generated meshes defined on a region.

Collaboration diagram for TYPES::GENERATED\_MESHES\_TYPE:

### Public Attributes

- INTEGER(INTG) [NUMBER\\_OF\\_GENERATED\\_MESHES](#)  
*The number of generated meshes defined.*
- TYPE([GENERATED\\_MESH\\_PTR\\_TYPE](#)), pointer [GENERATED\\_MESHES](#)  
*The array of pointers to the generated meshes.*

### 7.266.1 Detailed Description

Contains information on the generated meshes defined on a region.

Definition at line 348 of file types.f90.

### 7.266.2 Member Data Documentation

#### 7.266.2.1 TYPE(GENERATED\_MESH\_PTR\_TYPE),pointer TYPES::GENERATED\_MESHES\_TYPE::GENERATED\_MESHES

The array of pointers to the generated meshes.

Definition at line 350 of file types.f90.

#### 7.266.2.2 INTEGER(INTG) TYPES::GENERATED\_MESHES\_TYPE::NUMBER\_OF\_GENERATED\_MESHES

The number of generated meshes defined.

Definition at line 349 of file types.f90.

## 7.267 TYPES::JACOBIAN\_COL\_TO\_SOLVER\_COLS\_MAP\_- TYPE Struct Reference

### Public Attributes

- INTEGER(INTG) [NUMBER\\_OF\\_SOLVER\\_COLS](#)  
*The number of solver cols this Jacobian column is mapped to.*
- INTEGER(INTG), allocatable [SOLVER\\_COLS](#)  
*SOLVER\_COLS(i). The solver column number for the i'th solver column that this column is mapped to.*
- REAL(DP), allocatable [COUPLING\\_COEFFICIENTS](#)  
*COUPLING\_COEFFICIENTS(i). The coupling coefficients for the i'th solver column that this column is mapped to.*

### 7.267.1 Detailed Description

Definition at line 1490 of file types.f90.

### 7.267.2 Member Data Documentation

#### 7.267.2.1 REAL(DP),allocatable TYPES::JACOBIAN\_COL\_TO\_SOLVER\_COLS\_MAP\_- TYPE::COUPLING\_COEFFICIENTS

COUPLING\_COEFFICIENTS(i). The coupling coefficients for the i'th solver column that this column is mapped to.

Definition at line 1493 of file types.f90.

#### 7.267.2.2 INTEGER(INTG) TYPES::JACOBIAN\_COL\_TO\_SOLVER\_COLS\_MAP\_- TYPE::NUMBER\_OF\_SOLVER\_COLS

The number of solver cols this Jacobian column is mapped to.

Definition at line 1491 of file types.f90.

#### 7.267.2.3 INTEGER(INTG),allocatable TYPES::JACOBIAN\_COL\_TO\_SOLVER\_COLS\_- MAP\_TYPE::SOLVER\_COLS

SOLVER\_COLS(i). The solver column number for the i'th solver column that this column is mapped to.

Definition at line 1492 of file types.f90.

## 7.268 TYPES::JACOBIAN\_TO\_SOLVER\_MAP\_TYPE Struct Reference

Collaboration diagram for TYPES::JACOBIAN\_TO\_SOLVER\_MAP\_TYPE:

### Public Attributes

- INTEGER(INTG) **SOLVER\_MATRIX\_NUMBER**  
*The solver matrix number being mapped.*
- TYPE(EQUATIONS\_JACOBIAN\_TYPE), pointer **JACOBIAN\_MATRIX**  
*A pointer to the Jacobian matrix being mapped.*
- TYPE(SOLVER\_MATRIX\_TYPE), pointer **SOLVER\_MATRIX**  
*A pointer to the solver matrix being mapped.*
- TYPE(JACOBIAN\_COL\_TO\_SOLVER\_COLS\_MAP\_TYPE), allocatable **JACOBIAN\_COL\_-  
SOLVER\_COLS\_MAP**  
*JACOBIAN\_COL\_SOLVER\_COL\_MAP(column\_idx). The mapping from the column\_idx'th column of the Jacobian matrix to the solver matrix columns.*

### 7.268.1 Detailed Description

Definition at line 1496 of file types.f90.

### 7.268.2 Member Data Documentation

#### 7.268.2.1 TYPE(JACOBIAN\_COL\_TO\_SOLVER\_COLS\_MAP\_TYPE),allocatable TYPES::JACOBIAN\_TO\_SOLVER\_MAP\_TYPE::JACOBIAN\_COL\_SOLVER\_- COLS\_MAP

JACOBIAN\_COL\_SOLVER\_COL\_MAP(column\_idx). The mapping from the column\_idx'th column of the Jacobian matrix to the solver matrix columns.

Definition at line 1500 of file types.f90.

#### 7.268.2.2 TYPE(EQUATIONS\_JACOBIAN\_TYPE),pointer TYPES::JACOBIAN\_TO\_- SOLVER\_MAP\_TYPE::JACOBIAN\_MATRIX

A pointer to the Jacobian matrix being mapped.

Definition at line 1498 of file types.f90.

#### 7.268.2.3 TYPE(SOLVER\_MATRIX\_TYPE),pointer TYPES::JACOBIAN\_TO\_SOLVER\_- MAP\_TYPE::SOLVER\_MATRIX

A pointer to the solver matrix being mapped.

Definition at line 1499 of file types.f90.

**7.268.2.4 INTEGER(INTG) TYPES::JACOBIAN\_TO\_SOLVER\_MAP\_TYPE::SOLVER\_-  
MATRIX\_NUMBER**

The solver matrix number being mapped.

Definition at line 1497 of file types.f90.

## 7.269 TYPES::LINEAR\_DIRECT\_SOLVER\_TYPE Struct Reference

Contains information for a direct linear solver.

Collaboration diagram for TYPES::LINEAR\_DIRECT\_SOLVER\_TYPE:

### Public Attributes

- TYPE([LINEAR\\_SOLVER\\_TYPE](#)), pointer [LINEAR\\_SOLVER](#)  
*A pointer to the linear solver.*
- INTEGER(INTG) [SOLVER\\_LIBRARY](#)  
*The library type for the linear solver.*
- INTEGER(INTG) [DIRECT\\_SOLVER\\_TYPE](#)  
*The type of direct linear solver.*

### 7.269.1 Detailed Description

Contains information for a direct linear solver.

Definition at line 1374 of file types.f90.

### 7.269.2 Member Data Documentation

#### 7.269.2.1 INTEGER(INTG) TYPES::LINEAR\_DIRECT\_SOLVER\_TYPE::DIRECT\_- SOLVER\_TYPE

The type of direct linear solver.

Definition at line 1377 of file types.f90.

#### 7.269.2.2 TYPE([LINEAR\\_SOLVER\\_TYPE](#)),pointer TYPES::LINEAR\_DIRECT\_SOLVER\_- TYPE::[LINEAR\\_SOLVER](#)

A pointer to the linear solver.

Definition at line 1375 of file types.f90.

#### 7.269.2.3 INTEGER(INTG) TYPES::LINEAR\_DIRECT\_SOLVER\_TYPE::SOLVER\_LIBRARY

The library type for the linear solver.

#### See also:

[SOLVER\\_ROUTINES::SolverLibraries](#),[SOLVER\\_ROUTINES](#)

Definition at line 1376 of file types.f90.

## 7.270 TYPES::LINEAR\_ITERATIVE\_SOLVER\_TYPE Struct Reference

Contains information for a direct linear solver.

Collaboration diagram for TYPES::LINEAR\_ITERATIVE\_SOLVER\_TYPE:

### Public Attributes

- TYPE([LINEAR\\_SOLVER\\_TYPE](#)), pointer [LINEAR\\_SOLVER](#)  
*A pointer to the linear solver.*
- INTEGER(INTG) [SOLVER\\_LIBRARY](#)  
*The library type for the linear solver.*
- INTEGER(INTG) [ITERATIVE\\_SOLVER\\_TYPE](#)  
*The type of iterative solver.*
- INTEGER(INTG) [ITERATIVE\\_PRECONDITIONER\\_TYPE](#)  
*The type of iterative preconditioner.*
- INTEGER(INTG) [MAXIMUM\\_NUMBER\\_OF\\_ITERATIONS](#)  
*The maximum number of iterations.*
- REAL(DP) [RELATIVE\\_TOLERANCE](#)  
*The relative tolerance between the rhs and residual norm.*
- REAL(DP) [ABSOLUTE\\_TOLERANCE](#)  
*The absolute tolerance of the residual norm.*
- REAL(DP) [DIVERGENCE\\_TOLERANCE](#)  
*The absolute tolerance of the residual norm.*
- TYPE([PETSC\\_PC\\_TYPE](#)) [PC](#)  
*The PETSc preconditioner object.*
- TYPE([PETSC\\_KSP\\_TYPE](#)) [KSP](#)  
*The PETSc solver object.*

### 7.270.1 Detailed Description

Contains information for a direct linear solver.

Definition at line 1381 of file types.f90.

## 7.270.2 Member Data Documentation

### 7.270.2.1 REAL(DP) TYPES::LINEAR\_ITERATIVE\_SOLVER\_TYPE::ABSOLUTE\_TOLERANCE

The absolute tolerance of the residual norm.

Definition at line 1388 of file types.f90.

### 7.270.2.2 REAL(DP) TYPES::LINEAR\_ITERATIVE\_SOLVER\_TYPE::DIVERGENCE\_TOLERANCE

The absolute tolerance of the residual norm.

Definition at line 1389 of file types.f90.

### 7.270.2.3 INTEGER(INTG) TYPES::LINEAR\_ITERATIVE\_SOLVER\_TYPE::ITERATIVE\_PRECONDITIONER\_TYPE

The type of iterative preconditioner.

Definition at line 1385 of file types.f90.

### 7.270.2.4 INTEGER(INTG) TYPES::LINEAR\_ITERATIVE\_SOLVER\_TYPE::ITERATIVE\_SOLVER\_TYPE

The type of iterative solver.

Definition at line 1384 of file types.f90.

### 7.270.2.5 TYPE(PETSC\_KSP\_TYPE) TYPES::LINEAR\_ITERATIVE\_SOLVER\_TYPE::KSP

The PETSc solver object.

Definition at line 1391 of file types.f90.

### 7.270.2.6 TYPE(LINEAR\_SOLVER\_TYPE),pointer TYPES::LINEAR\_ITERATIVE\_SOLVER\_TYPE::LINEAR\_SOLVER

A pointer to the linear solver.

Definition at line 1382 of file types.f90.

### 7.270.2.7 INTEGER(INTG) TYPES::LINEAR\_ITERATIVE\_SOLVER\_TYPE::MAXIMUM\_NUMBER\_OF\_ITERATIONS

The maximum number of iterations.

Definition at line 1386 of file types.f90.

**7.270.2.8 TYPE(PETSC\_PC\_TYPE) TYPES::LINEAR\_ITERATIVE\_SOLVER\_TYPE::PC**

The PETSc preconditioner object.

Definition at line 1390 of file types.f90.

**7.270.2.9 REAL(DP) TYPES::LINEAR\_ITERATIVE\_SOLVER\_TYPE::RELATIVE\_TOLERANCE**

The relative tolerance between the rhs and residual norm.

Definition at line 1387 of file types.f90.

**7.270.2.10 INTEGER(INTG) TYPES::LINEAR\_ITERATIVE\_SOLVER\_TYPE::SOLVER\_LIBRARY**

The library type for the linear solver.

**See also:**

[SOLVER\\_ROUTINES::SolverLibraries](#), [SOLVER\\_ROUTINES](#)

Definition at line 1383 of file types.f90.

## 7.271 TYPES::LINEAR\_SOLVER\_TYPE Struct Reference

Contains information for a linear solver.

Collaboration diagram for TYPES::LINEAR\_SOLVER\_TYPE:

### Public Attributes

- TYPE([SOLVER\\_TYPE](#)), pointer [SOLVER](#)  
*A pointer to the solver.*
- INTEGER(INTG) [LINEAR\\_SOLVE\\_TYPE](#)  
*The type of linear solver.*
- TYPE([LINEAR\\_DIRECT\\_SOLVER\\_TYPE](#)), pointer [DIRECT\\_SOLVER](#)  
*A pointer to the direct solver information.*
- TYPE([LINEAR\\_ITERATIVE\\_SOLVER\\_TYPE](#)), pointer [ITERATIVE\\_SOLVER](#)  
*A pointer to the iterative solver information.*

### 7.271.1 Detailed Description

Contains information for a linear solver.

Definition at line 1395 of file types.f90.

### 7.271.2 Member Data Documentation

#### 7.271.2.1 TYPE([LINEAR\\_DIRECT\\_SOLVER\\_TYPE](#)),pointer [TYPES::LINEAR\\_SOLVER\\_TYPE::DIRECT\\_SOLVER](#)

A pointer to the direct solver information.

Definition at line 1398 of file types.f90.

#### 7.271.2.2 TYPE([LINEAR\\_ITERATIVE\\_SOLVER\\_TYPE](#)),pointer [TYPES::LINEAR\\_SOLVER\\_TYPE::ITERATIVE\\_SOLVER](#)

A pointer to the iterative solver information.

Definition at line 1399 of file types.f90.

#### 7.271.2.3 INTEGER(INTG) [TYPES::LINEAR\\_SOLVER\\_TYPE::LINEAR\\_SOLVE\\_TYPE](#)

The type of linear solver.

#### See also:

[SOLVER\\_ROUTINES::LinearSolverTypes](#),[SOLVER\\_ROUTINES](#)

Definition at line 1397 of file types.f90.

**7.271.2.4 TYPE(SOLVER\_TYPE),pointer TYPES::LINEAR\_SOLVER\_TYPE::SOLVER**

A pointer to the solver.

Definition at line 1396 of file types.f90.

## 7.272 TYPES::MATRIX\_TYPE Struct Reference

Contains information for a matrix.

### Public Attributes

- INTEGER(INTG) **ID**  
*The ID of the matrix.*
- LOGICAL **MATRIX\_FINISHED**  
*Is .TRUE. if the matrix has finished being created, .FALSE. if not.*
- INTEGER(INTG) **M**  
*The number of rows in the matrix.*
- INTEGER(INTG) **N**  
*The number of columns in the matrix.*
- INTEGER(INTG) **MAX\_M**  
*The maximum number of columns in the matrix storage.*
- INTEGER(INTG) **MAX\_N**  
*The maximum number of rows in the matrix storage.*
- INTEGER(INTG) **DATA\_TYPE**  
*The data type of the matrix.*
- INTEGER(INTG) **STORAGE\_TYPE**  
*The storage type of the matrix.*
- INTEGER(INTG) **NUMBER\_NON\_ZEROS**  
*The number of non-zero elements in the matrix.*
- INTEGER(INTG) **SIZE**  
*The size of the data arrays.*
- INTEGER(INTG) **MAXIMUM\_COLUMN\_INDICES\_PER\_ROW**  
*The maximum number of column indices for the rows.*
- INTEGER(INTG), allocatable **ROW\_INDICES**  
*ROW\_INDICES(i). The row indices for the matrix storage scheme.*
- INTEGER(INTG), allocatable **COLUMN\_INDICES**  
*COLUMN\_INDICES(i). The column indices for the matrix storage scheme.*
- INTEGER(INTG), allocatable **DATA\_INTG**  
*DATA\_INTG(i). The integer data for an integer matrix. The i'th component contains the data for the i'th matrix data stored on the domain.*

- REAL(SP), allocatable **DATA\_SP**

*DATA\_SP(i).* The real data for a single precision matrix. The i'th component contains the data for the i'th matrix data stored on the domain.

- REAL(DP), allocatable **DATA\_DP**

*DATA\_DP(i).* The real data for a double precision matrix. The i'th component contains the data for the i'th matrix data stored on the domain.

- LOGICAL, allocatable **DATA\_L**

*DATA\_L(i).* The logical data for a logical matrix. The i'th component contains the data for the i'th matrix data stored on the domain.

### 7.272.1 Detailed Description

Contains information for a matrix.

Definition at line 582 of file types.f90.

### 7.272.2 Member Data Documentation

#### 7.272.2.1 INTEGER(INTG),allocatable TYPES::MATRIX\_TYPE::COLUMN\_INDICES

COLUMN\_INDICES(i). The column indices for the matrix storage scheme.

See also:

MATRIX\_VECTOR\_MatrixStorageStructures

Definition at line 595 of file types.f90.

#### 7.272.2.2 REAL(DP),allocatable TYPES::MATRIX\_TYPE::DATA\_DP

DATA\_DP(i). The real data for a double precision matrix. The i'th component contains the data for the i'th matrix data stored on the domain.

Definition at line 598 of file types.f90.

#### 7.272.2.3 INTEGER(INTG),allocatable TYPES::MATRIX\_TYPE::DATA\_INTG

DATA\_INTG(i). The integer data for an integer matrix. The i'th component contains the data for the i'th matrix data stored on the domain.

Definition at line 596 of file types.f90.

#### 7.272.2.4 LOGICAL,allocatable TYPES::MATRIX\_TYPE::DATA\_L

DATA\_L(i). The logical data for a logical matrix. The i'th component contains the data for the i'th matrix data stored on the domain.

Definition at line 599 of file types.f90.

**7.272.2.5 REAL(SP),allocatable TYPES::MATRIX\_TYPE::DATA\_SP**

DATA\_SP(i). The real data for a single precision matrix. The i'th component contains the data for the i'th matrix data stored on the domain.

Definition at line 597 of file types.f90.

**7.272.2.6 INTEGER(INTG) TYPES::MATRIX\_TYPE::DATA\_TYPE**

The data type of the matrix.

See also:

[MATRIX\\_VECTOR::DataTypes](#)

Definition at line 589 of file types.f90.

**7.272.2.7 INTEGER(INTG) TYPES::MATRIX\_TYPE::ID**

The ID of the matrix.

Definition at line 583 of file types.f90.

**7.272.2.8 INTEGER(INTG) TYPES::MATRIX\_TYPE::M**

The number of rows in the matrix.

Definition at line 585 of file types.f90.

**7.272.2.9 LOGICAL TYPES::MATRIX\_TYPE::MATRIX\_FINISHED**

Is .TRUE. if the matrix has finished being created, .FALSE. if not.

Definition at line 584 of file types.f90.

**7.272.2.10 INTEGER(INTG) TYPES::MATRIX\_TYPE::MAX\_M**

The maximum number of columns in the matrix storage.

Definition at line 587 of file types.f90.

**7.272.2.11 INTEGER(INTG) TYPES::MATRIX\_TYPE::MAX\_N**

The maximum number of rows in the matrix storage.

Definition at line 588 of file types.f90.

**7.272.2.12 INTEGER(INTG) TYPES::MATRIX\_TYPE::MAXIMUM\_COLUMN\_INDICES\_-  
PER\_ROW**

The maximum number of column indicies for the rows.

Definition at line 593 of file types.f90.

**7.272.2.13 INTEGER(INTG) TYPES::MATRIX\_TYPE::N**

The number of columns in the matrix.

Definition at line 586 of file types.f90.

**7.272.2.14 INTEGER(INTG) TYPES::MATRIX\_TYPE::NUMBER\_NON\_ZEROS**

The number of non-zero elements in the matrix.

Definition at line 591 of file types.f90.

**7.272.2.15 INTEGER(INTG),allocatable TYPES::MATRIX\_TYPE::ROW\_INDICES**

ROW\_INDICES(i). The row indices for the matrix storage scheme.

**See also:**

[MATRIX\\_VECTOR\\_MatrixStorageStructures](#)

Definition at line 594 of file types.f90.

**7.272.2.16 INTEGER(INTG) TYPES::MATRIX\_TYPE::SIZE**

The size of the data arrays.

Definition at line 592 of file types.f90.

**7.272.2.17 INTEGER(INTG) TYPES::MATRIX\_TYPE::STORAGE\_TYPE**

The storage type of the matrix.

**See also:**

[MATRIX\\_VECTOR::StorageTypes](#)

Definition at line 590 of file types.f90.

## 7.273 TYPES::MESH\_DOFS\_TYPE Struct Reference

Contains information on the dofs for a mesh.

Collaboration diagram for TYPES::MESH\_DOFS\_TYPE:

### Public Attributes

- TYPE(MESH\_TYPE), pointer MESH  
*A pointer to the mesh.*
- INTEGER(INTG) NUMBER\_OF\_DOFS  
*The number of dofs in the mesh.*

### 7.273.1 Detailed Description

Contains information on the dofs for a mesh.

Definition at line 226 of file types.f90.

### 7.273.2 Member Data Documentation

#### 7.273.2.1 TYPE(MESH\_TYPE),pointer TYPES::MESH\_DOFS\_TYPE::MESH

A pointer to the mesh.

Definition at line 227 of file types.f90.

#### 7.273.2.2 INTEGER(INTG) TYPES::MESH\_DOFS\_TYPE::NUMBER\_OF\_DOFS

The number of dofs in the mesh.

Definition at line 228 of file types.f90.

## 7.274 TYPES::MESH\_ELEMENT\_TYPE Struct Reference

Contains the information for an element in a mesh.

Collaboration diagram for TYPES::MESH\_ELEMENT\_TYPE:

### Public Attributes

- INTEGER(INTG) [GLOBAL\\_NUMBER](#)  
*The global element number in the mesh.*
- INTEGER(INTG) [USER\\_NUMBER](#)  
*The corresponding user number for the element.*
- TYPE([BASIS\\_TYPE](#)), pointer [BASIS](#)  
*A pointer to the basis function for the element.*
- INTEGER(INTG), allocatable [GLOBAL\\_ELEMENT\\_NODES](#)  
*GLOBAL\_ELEMENT\_NODES(nn). The global node number in the mesh of the nn'th local node in the element. Old CMISS name NPNE(nn,nbf,ne).*
- INTEGER(INTG), allocatable [USER\\_ELEMENT\\_NODES](#)  
*USER\_ELEMENT\_NODES(nn). The user node number in the mesh of the nn'th local node in the element. Old CMISS name NPNE(nn,nbf,ne).*
- INTEGER(INTG), allocatable [NUMBER\\_OF\\_ADJACENT\\_ELEMENTS](#)  
*NUMBER\_OF\_ADJACENT\_ELEMENTS(-ni:ni). The number of elements adjacent to this element in the ni'th xi direction. Note that -ni gives the adjacent element before the element in the ni'th direction and +ni gives the adjacent element after the element in the ni'th direction. The ni=0 index should be 1 for the current element. Old CMISS name NXI(-ni:ni,0:nei,ne).*
- INTEGER(INTG), allocatable [ADJACENT\\_ELEMENTS](#)  
*ADJACENT\_ELEMENTS(nei,-ni:ni). The local element numbers of the elements adjacent to this element in the ni'th xi direction. Note that -ni gives the adjacent elements before the element in the ni'th direction and +ni gives the adjacent elements after the element in the ni'th direction. The ni=0 index should give the current element number. Old CMISS name NXI(-ni:ni,0:nei,ne).*

### 7.274.1 Detailed Description

Contains the information for an element in a mesh.

Definition at line 232 of file types.f90.

### 7.274.2 Member Data Documentation

#### 7.274.2.1 INTEGER(INTG),allocatable TYPES::MESH\_ELEMENT\_TYPE::ADJACENT\_ELEMENTS

ADJACENT\_ELEMENTS(nei,-ni:ni). The local element numbers of the elements adjacent to this element in the ni'th xi direction. Note that -ni gives the adjacent elements before the element in the ni'th direction

and  $+ni$  gives the adjacent elements after the element in the  $ni$ 'th direction. The  $ni=0$  index should give the current element number. Old [CMISS](#) name `NXI(-ni:ni,0:nei,ne)`.

Definition at line 239 of file types.f90.

#### **7.274.2.2 TYPE(BASIS\_TYPE),pointer TYPES::MESH\_ELEMENT\_TYPE::BASIS**

A pointer to the basis function for the element.

Definition at line 235 of file types.f90.

#### **7.274.2.3 INTEGER(INTG),allocatable TYPES::MESH\_ELEMENT\_TYPE::GLOBAL\_ELEMENT\_NODES**

`GLOBAL_ELEMENT_NODES(nn)`. The global node number in the mesh of the  $nn$ 'th local node in the element. Old [CMISS](#) name `NPNE(nn,nbf,ne)`.

Definition at line 236 of file types.f90.

#### **7.274.2.4 INTEGER(INTG) TYPES::MESH\_ELEMENT\_TYPE::GLOBAL\_NUMBER**

The global element number in the mesh.

Definition at line 233 of file types.f90.

#### **7.274.2.5 INTEGER(INTG),allocatable TYPES::MESH\_ELEMENT\_TYPE::NUMBER\_OF\_ADJACENT\_ELEMENTS**

`NUMBER_OF_ADJACENT_ELEMENTS(-ni:ni)`. The number of elements adjacent to this element in the  $ni$ 'th xi direction. Note that  $-ni$  gives the adjacent element before the element in the  $ni$ 'th direction and  $+ni$  gives the adjacent element after the element in the  $ni$ 'th direction. The  $ni=0$  index should be 1 for the current element. Old [CMISS](#) name `NXI(-ni:ni,0:nei,ne)`.

Definition at line 238 of file types.f90.

#### **7.274.2.6 INTEGER(INTG),allocatable TYPES::MESH\_ELEMENT\_TYPE::USER\_ELEMENT\_NODES**

`USER_ELEMENT_NODES(nn)`. The user node number in the mesh of the  $nn$ 'th local node in the element. Old [CMISS](#) name `NPNE(nn,nbf,ne)`.

Definition at line 237 of file types.f90.

#### **7.274.2.7 INTEGER(INTG) TYPES::MESH\_ELEMENT\_TYPE::USER\_NUMBER**

The corresponding user number for the element.

Definition at line 234 of file types.f90.

## 7.275 TYPES::MESH\_ELEMENTS\_TYPE Struct Reference

Contains the information for the elements of a mesh.

Collaboration diagram for TYPES::MESH\_ELEMENTS\_TYPE:

### Public Attributes

- TYPE(MESH\_TYPE), pointer MESH

*The pointer to the mesh for the elements information.*

- INTEGER(INTG) NUMBER\_OF\_ELEMENTS

*The number of elements in the mesh.*

- LOGICAL ELEMENTS\_FINISHED

*Is .TRUE. if the mesh elements have finished being created, .FALSE. if not.*

- TYPE(MESH\_ELEMENT\_TYPE), pointer ELEMENTS

*ELEMENTS(ne). The pointer to the array of information for the elements of this mesh. ELEMENTS(ne) contains the information for the ne'th global element of the mesh.*

### 7.275.1 Detailed Description

Contains the information for the elements of a mesh.

Definition at line 243 of file types.f90.

### 7.275.2 Member Data Documentation

#### 7.275.2.1 TYPE(MESH\_ELEMENT\_TYPE),pointer TYPES::MESH\_ELEMENTS\_TYPE::ELEMENTS

ELEMENTS(ne). The pointer to the array of information for the elements of this mesh. ELEMENTS(ne) contains the information for the ne'th global element of the mesh.

#### Todo

Should this be allocatable.

Definition at line 247 of file types.f90.

#### 7.275.2.2 LOGICAL TYPES::MESH\_ELEMENTS\_TYPE::ELEMENTS\_FINISHED

Is .TRUE. if the mesh elements have finished being created, .FALSE. if not.

Definition at line 246 of file types.f90.

**7.275.2.3 TYPE(MESH\_TYPE),pointer TYPES::MESH\_ELEMENTS\_TYPE::MESH**

The pointer to the mesh for the elements information.

Definition at line 244 of file types.f90.

**7.275.2.4 INTEGER(INTG) TYPES::MESH\_ELEMENTS\_TYPE::NUMBER\_OF\_ELEMENTS**

The number of elements in the mesh.

Definition at line 245 of file types.f90.

## 7.276 TYPES::MESH\_NODE\_TYPE Struct Reference

Contains the topology information for a global node of a mesh.

### Public Attributes

- INTEGER(INTG) [GLOBAL\\_NUMBER](#)  
*The global node number in the mesh.*
- INTEGER(INTG) [USER\\_NUMBER](#)  
*The corresponding user number for the node.*
- INTEGER(INTG) [NUMBER\\_OF\\_DERIVATIVES](#)  
*The number of global derivatives at the node for the mesh. Old CMISS name NKT(nj,np).*
- INTEGER(INTG), allocatable [PARTIAL\\_DERIVATIVE\\_INDEX](#)  
*PARTIAL\_DERIVATIVE\_INDEX(nk). The partial derivative index (nu) of the nk'th global derivative for the node. Old CMISS name NUNK(nk,nj,np).*
- INTEGER(INTG), allocatable [DOF\\_INDEX](#)  
*DOF\_INDEX(nk). The global dof derivative index (ny) in the domain of the nk'th global derivative for the node.*
- INTEGER(INTG) [NUMBER\\_OF\\_SURROUNDING\\_ELEMENTS](#)  
*The number of elements surrounding the node in the mesh. Old CMISS name NENP(np,0,0:nr).*
- INTEGER(INTG), pointer [SURROUNDING\\_ELEMENTS](#)  
*SURROUNDING\_ELEMENTS(nep). The global element number of the nep'th element that is surrounding the node. Old CMISS name NENP(np,nep,0:nr).*

### 7.276.1 Detailed Description

Contains the topology information for a global node of a mesh.

Definition at line 251 of file types.f90.

### 7.276.2 Member Data Documentation

#### 7.276.2.1 INTEGER(INTG),allocatable TYPES::MESH\_NODE\_TYPE::DOF\_INDEX

DOF\_INDEX(nk). The global dof derivative index (ny) in the domain of the nk'th global derivative for the node.

Definition at line 256 of file types.f90.

#### 7.276.2.2 INTEGER(INTG) TYPES::MESH\_NODE\_TYPE::GLOBAL\_NUMBER

The global node number in the mesh.

Definition at line 252 of file types.f90.

**7.276.2.3 INTEGER(INTG) TYPES::MESH\_NODE\_TYPE::NUMBER\_OF\_DERIVATIVES**

The number of global derivatives at the node for the mesh. Old [CMISS](#) name NKT(nj,np).

Definition at line 254 of file types.f90.

**7.276.2.4 INTEGER(INTG) TYPES::MESH\_NODE\_TYPE::NUMBER\_OF\_SURROUNDING\_ELEMENTS**

The number of elements surrounding the node in the mesh. Old [CMISS](#) name NENP(np,0,0:nr).

Definition at line 257 of file types.f90.

**7.276.2.5 INTEGER(INTG),allocatable TYPES::MESH\_NODE\_TYPE::PARTIAL\_DERIVATIVE\_INDEX**

PARTIAL\_DERIVATIVE\_INDEX(nk). The partial derivative index (nu) of the nk'th global derivative for the node. Old [CMISS](#) name NUNK(nk,nj,np).

Definition at line 255 of file types.f90.

**7.276.2.6 INTEGER(INTG),pointer TYPES::MESH\_NODE\_TYPE::SURROUNDING\_ELEMENTS**

SURROUNDING\_ELEMENTS(nep). The global element number of the nep'th element that is surrounding the node. Old [CMISS](#) name NENP(np,nep,0:nr).

**Todo**

Change this to allocatable.

Definition at line 258 of file types.f90.

**7.276.2.7 INTEGER(INTG) TYPES::MESH\_NODE\_TYPE::USER\_NUMBER**

The corresponding user number for the node.

Definition at line 253 of file types.f90.

## 7.277 TYPES::MESH\_NODES\_TYPE Struct Reference

Contains the information for the nodes of a mesh.

Collaboration diagram for TYPES::MESH\_NODES\_TYPE:

### Public Attributes

- TYPE(MESH\_TYPE), pointer MESH  
*The pointer to the mesh for this nodes information.*
- INTEGER(INTG) NUMBER\_OF\_NODES  
*The number of nodes in the mesh.*
- TYPE(MESH\_NODE\_TYPE), pointer NODES  
*NODES(np). The pointer to the array of topology information for the nodes of the mesh. NODES(np) contains the topological information for the np'th global node of the mesh.*

### 7.277.1 Detailed Description

Contains the information for the nodes of a mesh.

Definition at line 262 of file types.f90.

### 7.277.2 Member Data Documentation

#### 7.277.2.1 TYPE(MESH\_TYPE),pointer TYPES::MESH\_NODES\_TYPE::MESH

The pointer to the mesh for this nodes information.

Definition at line 263 of file types.f90.

#### 7.277.2.2 TYPE(MESH\_NODE\_TYPE),pointer TYPES::MESH\_NODES\_TYPE::NODES

NODES(np). The pointer to the array of topology information for the nodes of the mesh. NODES(np) contains the topological information for the np'th global node of the mesh.

#### Todo

Should this be allocatable???

Definition at line 265 of file types.f90.

#### 7.277.2.3 INTEGER(INTG) TYPES::MESH\_NODES\_TYPE::NUMBER\_OF\_NODES

The number of nodes in the mesh.

Definition at line 264 of file types.f90.

## 7.278 TYPES::MESH\_PTR\_TYPE Struct Reference

A buffer type to allow for an array of pointers to a MESH\_TYPE.

Collaboration diagram for TYPES::MESH\_PTR\_TYPE:

### Public Attributes

- TYPE(MESH\_TYPE), pointer PTR

*The pointer to the mesh.*

### 7.278.1 Detailed Description

A buffer type to allow for an array of pointers to a MESH\_TYPE.

Definition at line 304 of file types.f90.

### 7.278.2 Member Data Documentation

#### 7.278.2.1 TYPE(MESH\_TYPE),pointer TYPES::MESH\_PTR\_TYPE::PTR

The pointer to the mesh.

Definition at line 305 of file types.f90.

## 7.279 TYPES::MESH\_TOPOLOGY\_PTR\_TYPE Struct Reference

A buffer type to allow for an array of pointers to a MESH\_TOPOLOGY\_TYPE.

Collaboration diagram for TYPES::MESH\_TOPOLOGY\_PTR\_TYPE:

### Public Attributes

- TYPE(MESH\_TOPOLOGY\_TYPE), pointer PTR

*The pointer to the mesh topology.*

#### 7.279.1 Detailed Description

A buffer type to allow for an array of pointers to a MESH\_TOPOLOGY\_TYPE.

Definition at line 278 of file types.f90.

#### 7.279.2 Member Data Documentation

##### 7.279.2.1 TYPE(MESH\_TOPOLOGY\_TYPE),pointer TYPES::MESH\_TOPOLOGY\_PTR\_- TYPE::PTR

The pointer to the mesh topology.

Definition at line 279 of file types.f90.

## 7.280 TYPES::MESH\_TOPOLOGY\_TYPE Struct Reference

Contains information on the (global) topology of a mesh.

Collaboration diagram for TYPES::MESH\_TOPOLOGY\_TYPE:

### Public Attributes

- TYPE(MESH\_TYPE), pointer MESH  
*Pointer to the parent mesh.*
- INTEGER(INTG) MESH\_COMPONENT\_NUMBER  
*The mesh component number for this mesh topology.*
- TYPE(MESH\_NODES\_TYPE), pointer NODES  
*Pointer to the nodes within the mesh topology.*
- TYPE(MESH\_ELEMENTS\_TYPE), pointer ELEMENTS  
*Pointer to the elements within the mesh topology.*
- TYPE(MESH\_DOFS\_TYPE), pointer DOFS  
*Pointer to the dofs within the mesh topology.*

### 7.280.1 Detailed Description

Contains information on the (global) topology of a mesh.

Definition at line 269 of file types.f90.

### 7.280.2 Member Data Documentation

#### 7.280.2.1 TYPE(MESH\_DOFS\_TYPE),pointer TYPES::MESH\_TOPOLOGY\_TYPE::DOFS

Pointer to the dofs within the mesh topology.

Definition at line 274 of file types.f90.

#### 7.280.2.2 TYPE(MESH\_ELEMENTS\_TYPE),pointer TYPES::MESH\_TOPOLOGY\_TYPE::ELEMENTS

Pointer to the elements within the mesh topology.

Definition at line 273 of file types.f90.

#### 7.280.2.3 TYPE(MESH\_TYPE),pointer TYPES::MESH\_TOPOLOGY\_TYPE::MESH

Pointer to the parent mesh.

Definition at line 270 of file types.f90.

**7.280.2.4 INTEGER(INTG) TYPES::MESH\_TOPOLOGY\_TYPE::MESH\_COMPONENT\_NUMBER**

The mesh component number for this mesh topology.

Definition at line 271 of file types.f90.

**7.280.2.5 TYPE(MESH\_NODES\_TYPE),pointer TYPES::MESH\_TOPOLOGY\_TYPE::NODES**

Pointer to the nodes within the mesh topology.

Definition at line 272 of file types.f90.

## 7.281 TYPES::MESH\_TYPE Struct Reference

Contains information on a mesh defined on a region.

Collaboration diagram for TYPES::MESH\_TYPE:

### Public Attributes

- INTEGER(INTG) [USER\\_NUMBER](#)  
*The user number of the mesh. The user number must be unique.*
- INTEGER(INTG) [GLOBAL\\_NUMBER](#)  
*The corresponding global number for the mesh.*
- LOGICAL [MESH\\_FINISHED](#)  
*Is .TRUE. if the mesh has finished being created, .FALSE. if not.*
- TYPE([MESHS\\_TYPE](#)), pointer [MESHS](#)  
*A pointer to the meshes for this mesh.*
- TYPE([REGION\\_TYPE](#)), pointer [REGION](#)  
*A pointer to the region containing this mesh.*
- TYPE([GENERATED\\_MESH\\_TYPE](#)), pointer [GENERATED\\_MESH](#)  
*A pointer to the generated mesh generate this mesh.*
- INTEGER(INTG) [NUMBER\\_OF\\_DIMENSIONS](#)  
*The number of dimensions ( $X_i$  directions) for this mesh.*
- INTEGER(INTG) [NUMBER\\_OF\\_COMPONENTS](#)  
*The number of mesh components in this mesh.*
- LOGICAL [MESH\\_EMBEDDED](#)  
*Is .TRUE. if the mesh is embedded in another mesh, .FALSE. if not.*
- TYPE([MESH\\_TYPE](#)), pointer [EMBEDDING\\_MESH](#)  
*If this mesh is embedded the pointer to the mesh that this mesh is embedded in. IF the mesh is not embedded the pointer is NULL.*
- INTEGER(INTG) [NUMBER\\_OF\\_EMBEDDED\\_MESHES](#)  
*The number of meshes that are embedded in this mesh.*
- TYPE([MESH\\_PTR\\_TYPE](#)), pointer [EMBEDDED\\_MESHES](#)  
*EMBEDDED\_MESHES(mesh\_idx). A pointer to the mesh\_idx'th mesh that is embedded in this mesh.*
- INTEGER(INTG) [NUMBER\\_OF\\_ELEMENTS](#)  
*The number of elements in the mesh.*
- INTEGER(INTG) [NUMBER\\_OF\\_FACES](#)  
*The number of faces in the mesh.*

- INTEGER(INTG) [NUMBER\\_OF\\_LINES](#)

*The number of lines in the mesh.*

- TYPE([MESH\\_TOPOLOGY\\_PTR\\_TYPE](#)), pointer [TOPOLOGY](#)

*TOPOLOGY(mesh\_component\_idx). A pointer to the topology mesh\_component\_idx'th mesh component.*

- TYPE([DECOMPOSITIONS\\_TYPE](#)), pointer [DECOMPOSITIONS](#)

*A pointer to the decompositions for this mesh.*

## 7.281.1 Detailed Description

Contains information on a mesh defined on a region.

Definition at line 283 of file types.f90.

## 7.281.2 Member Data Documentation

### 7.281.2.1 TYPE(DECOMPOSITIONS\_TYPE),pointer TYPES::MESH\_- TYPE::DECOMPOSITIONS

A pointer to the decompositions for this mesh.

Definition at line 300 of file types.f90.

### 7.281.2.2 TYPE(MESH\_PTR\_TYPE),pointer TYPES::MESH\_TYPE::EMBEDDED\_MESHES

EMBEDDED\_MESHES(mesh\_idx). A pointer to the mesh\_idx'th mesh that is embedded in this mesh.

Definition at line 295 of file types.f90.

### 7.281.2.3 TYPE(MESH\_TYPE),pointer TYPES::MESH\_TYPE::EMBEDDING\_MESH

If this mesh is embedded the pointer to the mesh that this mesh is embedded in. IF the mesh is not embedded the pointer is NULL.

Definition at line 293 of file types.f90.

### 7.281.2.4 TYPE(GENERATED\_MESH\_TYPE),pointer TYPES::MESH\_TYPE::GENERATED\_- MESH

A pointer to the generated mesh generate this mesh.

Definition at line 289 of file types.f90.

### 7.281.2.5 INTEGER(INTG) TYPES::MESH\_TYPE::GLOBAL\_NUMBER

The corresponding global number for the mesh.

Definition at line 285 of file types.f90.

**7.281.2.6 LOGICAL TYPES::MESH\_TYPE::MESH\_EMBEDDED**

Is .TRUE. if the mesh is embedded in another mesh, .FALSE. if not.

Definition at line 292 of file types.f90.

**7.281.2.7 LOGICAL TYPES::MESH\_TYPE::MESH\_FINISHED**

Is .TRUE. if the mesh has finished being created, .FALSE. if not.

Definition at line 286 of file types.f90.

**7.281.2.8 TYPE(MESHES\_TYPE),pointer TYPES::MESH\_TYPE::MESHES**

A pointer to the meshes for this mesh.

Definition at line 287 of file types.f90.

**7.281.2.9 INTEGER(INTG) TYPES::MESH\_TYPE::NUMBER\_OF\_COMPONENTS**

The number of mesh components in this mesh.

Definition at line 291 of file types.f90.

**7.281.2.10 INTEGER(INTG) TYPES::MESH\_TYPE::NUMBER\_OF\_DIMENSIONS**

The number of dimensions (Xi directions) for this mesh.

Definition at line 290 of file types.f90.

**7.281.2.11 INTEGER(INTG) TYPES::MESH\_TYPE::NUMBER\_OF\_ELEMENTS**

The number of elements in the mesh.

Definition at line 296 of file types.f90.

**7.281.2.12 INTEGER(INTG) TYPES::MESH\_TYPE::NUMBER\_OF\_EMBEDDED\_MESHES**

The number of meshes that are embedded in this mesh.

Definition at line 294 of file types.f90.

**7.281.2.13 INTEGER(INTG) TYPES::MESH\_TYPE::NUMBER\_OF\_FACES**

The number of faces in the mesh.

Definition at line 297 of file types.f90.

**7.281.2.14 INTEGER(INTG) TYPES::MESH\_TYPE::NUMBER\_OF\_LINES**

The number of lines in the mesh.

Definition at line 298 of file types.f90.

**7.281.2.15 TYPE(REGION\_TYPE),pointer TYPES::MESH\_TYPE::REGION**

A pointer to the region containing this mesh.

Definition at line 288 of file types.f90.

**7.281.2.16 TYPE(MESH\_TOPOLOGY\_PTR\_TYPE),pointer TYPES::MESH\_TYPE::TOPOLOGY**

TOPOLOGY(mesh\_component\_idx). A pointer to the topology mesh\_component\_idx'th mesh component.

Definition at line 299 of file types.f90.

**7.281.2.17 INTEGER(INTG) TYPES::MESH\_TYPE::USER\_NUMBER**

The user number of the mesh. The user number must be unique.

Definition at line 284 of file types.f90.

## 7.282 TYPES::MESHES\_TYPE Struct Reference

Contains information on the meshes defined on a region.

Collaboration diagram for TYPES::MESHES\_TYPE:

### Public Attributes

- TYPE([REGION\\_TYPE](#)), pointer [REGION](#)  
*A pointer to the region.*
- INTEGER(INTG) [NUMBER\\_OF\\_MESHES](#)  
*The number of meshes defined on the region.*
- TYPE([MESH\\_PTR\\_TYPE](#)), pointer [MESHES](#)  
*MESHES(meshes\_idx). The array of pointers to the meshes.*

### 7.282.1 Detailed Description

Contains information on the meshes defined on a region.

Definition at line 309 of file types.f90.

### 7.282.2 Member Data Documentation

#### 7.282.2.1 TYPE(MESH\_PTR\_TYPE),pointer TYPES::MESHES\_TYPE::MESHES

MESHES(meshes\_idx). The array of pointers to the meshes.

Definition at line 312 of file types.f90.

#### 7.282.2.2 INTEGER(INTG) TYPES::MESHES\_TYPE::NUMBER\_OF\_MESHES

The number of meshes defined on the region.

Definition at line 311 of file types.f90.

#### 7.282.2.3 TYPE(REGION\_TYPE),pointer TYPES::MESHES\_TYPE::REGION

A pointer to the region.

Definition at line 310 of file types.f90.

## 7.283 TYPES::NODE\_TYPE Struct Reference

Contains information about a node.

Collaboration diagram for TYPES::NODE\_TYPE:

### Public Attributes

- INTEGER(INTG) [GLOBAL\\_NUMBER](#)

*The global number of node.*

- INTEGER(INTG) [USER\\_NUMBER](#)

*The user defined number of node.*

- TYPE([VARYING\\_STRING](#)) [LABEL](#)

*A string label for the node.*

- REAL(DP), allocatable [INITIAL\\_POSITION](#)

*INITIAL\_POSITION(nj). The initial position of the node. Used to identify the position of the node before meshing (e.g., Delauny triangulation) as the geometric field has not been created when the node is created. The actual geometric coordinates for computation using the node should be taken from the geometric field which uses the node.*

### 7.283.1 Detailed Description

Contains information about a node.

Definition at line 203 of file types.f90.

### 7.283.2 Member Data Documentation

#### 7.283.2.1 INTEGER(INTG) TYPES::NODE\_TYPE::GLOBAL\_NUMBER

The global number of node.

Definition at line 204 of file types.f90.

#### 7.283.2.2 REAL(DP),allocatable TYPES::NODE\_TYPE::INITIAL\_POSITION

*INITIAL\_POSITION(nj). The initial position of the node. Used to identify the position of the node before meshing (e.g., Delauny triangulation) as the geometric field has not been created when the node is created. The actual geometric coordinates for computation using the node should be taken from the geometric field which uses the node.*

Definition at line 207 of file types.f90.

#### 7.283.2.3 TYPE(VARYING\_STRING) TYPES::NODE\_TYPE::LABEL

A string label for the node.

Definition at line 206 of file types.f90.

**7.283.2.4 INTEGER(INTG) TYPES::NODE\_TYPE::USER\_NUMBER**

The user defined number of node.

Definition at line 205 of file types.f90.

## 7.284 TYPES::NODES\_TYPE Struct Reference

Contains information on the nodes defined on a region.

Collaboration diagram for TYPES::NODES\_TYPE:

### Public Attributes

- TYPE(REGION\_TYPE), pointer REGION  
*A pointer to the region containing the nodes.*
- INTEGER(INTG) NUMBER\_OF\_NODES  
*The number of nodes defined on the region.*
- LOGICAL NODES\_FINISHED  
*Is .TRUE. if the nodes have finished being created, .FALSE. if not.*
- TYPE(NODE\_TYPE), pointer NODES  
*NODES(nodes\_idx). A pointer to the nodes.*
- TYPE(TREE\_TYPE), pointer NODE\_TREE  
*The tree for user to global node mapping.*

### 7.284.1 Detailed Description

Contains information on the nodes defined on a region.

Definition at line 211 of file types.f90.

### 7.284.2 Member Data Documentation

#### 7.284.2.1 TYPE(TREE\_TYPE),pointer TYPES::NODES\_TYPE::NODE\_TREE

The tree for user to global node mapping.

Definition at line 216 of file types.f90.

#### 7.284.2.2 TYPE(NODE\_TYPE),pointer TYPES::NODES\_TYPE::NODES

NODES(nodes\_idx). A pointer to the nodes.

#### Todo

Should this be allocatable?

Definition at line 215 of file types.f90.

**7.284.2.3 LOGICAL TYPES::NODES\_TYPE::NODES\_FINISHED**

Is .TRUE. if the nodes have finished being created, .FALSE. if not.

Definition at line 214 of file types.f90.

**7.284.2.4 INTEGER(INTG) TYPES::NODES\_TYPE::NUMBER\_OF\_NODES**

The number of nodes defined on the region.

Definition at line 213 of file types.f90.

**7.284.2.5 TYPE(REGION\_TYPE),pointer TYPES::NODES\_TYPE::REGION**

A pointer to the region containing the nodes.

Definition at line 212 of file types.f90.

## 7.285 TYPES::NONLINEAR\_LINESearch\_SOLVER\_TYPE Struct Reference

Contains information for a Newton line search nonlinear solver.

Collaboration diagram for TYPES::NONLINEAR\_LINESearch\_SOLVER\_TYPE:

### Public Attributes

- TYPE(NONLINEAR\_SOLVER\_TYPE), pointer NONLINEAR\_SOLVER  
*A pointer to the nonlinear solver.*
- INTEGER(INTG) SOLVER\_LIBRARY  
*The library type for the nonlinear solver.*
- INTEGER(INTG) JACOBIAN\_CALCULATION\_TYPE  
*The type of calculation used for the Jacobian.*
- INTEGER(INTG) LINESEARCH\_TYPE  
*The line search type.*
- REAL(DP) LINESEARCH\_ALPHA  
*The line search alpha.*
- REAL(DP) LINESEARCH\_MAXSTEP  
*The line search maximum step.*
- REAL(DP) LINESEARCH\_STEPTOLERANCE  
*The line search step tolerance.*
- TYPE(PETSC\_ISCOLORING\_TYPE) JACOBIAN\_ISCOLORING  
*The Jacobian matrix index set colouring.*
- TYPE(PETSC\_MATFDCOLORING\_TYPE) JACOBIAN\_FDCOLORING  
*The Jacobian matrix finite difference colouring.*
- TYPE(PETSC\_SNES\_TYPE) SNES  
*The PETSc nonlinear solver object.*

### 7.285.1 Detailed Description

Contains information for a Newton line search nonlinear solver.

Definition at line 1403 of file types.f90.

## 7.285.2 Member Data Documentation

### 7.285.2.1 INTEGER(INTG) TYPES::NONLINEAR\_LINESearch\_SOLVER\_- TYPE::JACOBIAN\_CALCULATION\_TYPE

The type of calculation used for the Jacobian.

See also:

[SOLVER\\_ROUTINES::JacobianCalculationTypes](#), [SOLVER\\_ROUTINES](#)

Definition at line 1406 of file types.f90.

### 7.285.2.2 TYPE(PETSC\_MATFDCOLORING\_TYPE) TYPES::NONLINEAR\_LINESearch\_- SOLVER\_TYPE::JACOBIAN\_FDCOLORING

The Jacobian matrix finite difference colouring.

Definition at line 1412 of file types.f90.

### 7.285.2.3 TYPE(PETSC\_ISCOLORING\_TYPE) TYPES::NONLINEAR\_LINESearch\_- SOLVER\_TYPE::JACOBIAN\_ISCOLORING

The Jacobian matrix index set colouring.

Definition at line 1411 of file types.f90.

### 7.285.2.4 REAL(DP) TYPES::NONLINEAR\_LINESearch\_SOLVER\_- TYPE::LINESEARCH\_ALPHA

The line search alpha.

Definition at line 1408 of file types.f90.

### 7.285.2.5 REAL(DP) TYPES::NONLINEAR\_LINESearch\_SOLVER\_- TYPE::LINESEARCH\_MAXSTEP

The line search maximum step.

Definition at line 1409 of file types.f90.

### 7.285.2.6 REAL(DP) TYPES::NONLINEAR\_LINESearch\_SOLVER\_- TYPE::LINESEARCH\_STEPTOLERANCE

The line search step tolerance.

Definition at line 1410 of file types.f90.

### 7.285.2.7 INTEGER(INTG) TYPES::NONLINEAR\_LINESearch\_SOLVER\_- TYPE::LINESEARCH\_TYPE

The line search type.

See also:

[SOLVER\\_ROUTINES::NonlinearLineSearchTypes](#),[SOLVER\\_ROUTINES](#)

Definition at line 1407 of file types.f90.

**7.285.2.8 TYPE(NONLINEAR\_SOLVER\_TYPE),pointer TYPES::NONLINEAR\_-  
LINESEARCH\_SOLVER\_TYPE::NONLINEAR\_SOLVER**

A pointer to the nonlinear solver.

Definition at line 1404 of file types.f90.

**7.285.2.9 TYPE(PETSC\_SNES\_TYPE) TYPES::NONLINEAR\_LINESEARCH\_SOLVER\_-  
TYPE::SNES**

The PETSc nonlinear solver object.

Definition at line 1413 of file types.f90.

**7.285.2.10 INTEGER(INTG) TYPES::NONLINEAR\_LINESEARCH\_SOLVER\_-  
TYPE::SOLVER\_LIBRARY**

The library type for the nonlinear solver.

See also:

[SOLVER\\_ROUTINES::SolverLibraries](#),[SOLVER\\_ROUTINES](#)

Definition at line 1405 of file types.f90.

## 7.286 TYPES::NONLINEAR\_SOLVER\_TYPE Struct Reference

Contains information for a nonlinear solver.

Collaboration diagram for TYPES::NONLINEAR\_SOLVER\_TYPE:

### Public Attributes

- TYPE(SOLVER\_TYPE), pointer SOLVER  
*A pointer to the problem\_solver.*
- INTEGER(INTG) NONLINEAR\_SOLVE\_TYPE  
*The type of nonlinear solver.*
- INTEGER(INTG) MAXIMUM\_NUMBER\_OF\_ITERATIONS  
*The maximum number of iterations.*
- INTEGER(INTG) MAXIMUM\_NUMBER\_OF\_FUNCTION\_EVALUATIONS  
*The maximum number of function evaluations.*
- REAL(DP) ABSOLUTE\_TOLERANCE  
*The tolerance between the absolute decrease between the solution norm and the initial guess.*
- REAL(DP) RELATIVE\_TOLERANCE  
*The tolerance between the relative decrease between the solution norm and the initial guess.*
- REAL(DP) SOLUTION\_TOLERANCE  
*The tolerance of the change in the norm of the solution.*
- TYPE(NONLINEAR\_LINESEARCH\_SOLVER\_TYPE), pointer LINESEARCH\_SOLVER  
*A pointer to the Newton line search solver information.*
- TYPE(NONLINEAR\_TRUSTREGION\_SOLVER\_TYPE), pointer TRUSTREGION\_SOLVER  
*A pointer to the Newton trust region solver information.*

### 7.286.1 Detailed Description

Contains information for a nonlinear solver.

Definition at line 1426 of file types.f90.

### 7.286.2 Member Data Documentation

#### 7.286.2.1 REAL(DP) TYPES::NONLINEAR\_SOLVER\_TYPE::ABSOLUTE\_TOLERANCE

The tolerance between the absolute decrease between the solution norm and the initial guess.

Definition at line 1431 of file types.f90.

**7.286.2.2 TYPE(NONLINEAR\_LINESEARCH\_SOLVER\_TYPE),pointer  
TYPES::NONLINEAR\_SOLVER\_TYPE::LINESEARCH\_SOLVER**

A pointer to the Newton line search solver information.

Definition at line 1434 of file types.f90.

**7.286.2.3 INTEGER(INTG) TYPES::NONLINEAR\_SOLVER\_TYPE::MAXIMUM\_NUMBER\_-  
OF\_FUNCTION\_EVALUATIONS**

The maximum number of function evaluations.

Definition at line 1430 of file types.f90.

**7.286.2.4 INTEGER(INTG) TYPES::NONLINEAR\_SOLVER\_TYPE::MAXIMUM\_NUMBER\_-  
OF\_ITERATIONS**

The maximum number of iterations.

Definition at line 1429 of file types.f90.

**7.286.2.5 INTEGER(INTG) TYPES::NONLINEAR\_SOLVER\_TYPE::NONLINEAR\_SOLVE\_-  
TYPE**

The type of nonlinear solver.

**See also:**

[SOLVER\\_ROUTINES::NonlinearSolverTypes,SOLVER\\_ROUTINES](#)

Definition at line 1428 of file types.f90.

**7.286.2.6 REAL(DP) TYPES::NONLINEAR\_SOLVER\_TYPE::RELATIVE\_TOLERANCE**

The tolerance between the relative decrease between the solution norm and the initial guess.

Definition at line 1432 of file types.f90.

**7.286.2.7 REAL(DP) TYPES::NONLINEAR\_SOLVER\_TYPE::SOLUTION\_TOLERANCE**

The tolerance of the change in the norm of the solution.

Definition at line 1433 of file types.f90.

**7.286.2.8 TYPE(SOLVER\_TYPE),pointer TYPES::NONLINEAR\_SOLVER\_TYPE::SOLVER**

A pointer to the problem\_solver.

Definition at line 1427 of file types.f90.

**7.286.2.9 TYPE(NONLINEAR\_TRUSTREGION\_SOLVER\_TYPE),pointer  
TYPES::NONLINEAR\_SOLVER\_TYPE::TRUSTREGION\_SOLVER**

A pointer to the Newton trust region solver information.

Definition at line 1435 of file types.f90.

## 7.287 TYPES::NONLINEAR\_TRUSTREGION\_SOLVER\_TYPE Struct Reference

Contains information for a Newton trust region nonlinear solver.

Collaboration diagram for TYPES::NONLINEAR\_TRUSTREGION\_SOLVER\_TYPE:

### Public Attributes

- TYPE(NONLINEAR\_SOLVER\_TYPE), pointer NONLINEAR\_SOLVER  
*A pointer to the nonlinear solver.*
- INTEGER(INTG) SOLVER\_LIBRARY  
*The library type for the nonlinear solver.*
- REAL(DP) TRUSTREGION\_TOLERANCE  
*The trust region tolerance.*
- REAL(DP) TRUSTREGION\_DELTA0  
*The trust region delta0.*
- TYPE(PETSC\_SNES\_TYPE) SNES  
*The PETSc nonlinear solver object.*

### 7.287.1 Detailed Description

Contains information for a Newton trust region nonlinear solver.

Definition at line 1417 of file types.f90.

### 7.287.2 Member Data Documentation

#### 7.287.2.1 TYPE(NONLINEAR\_SOLVER\_TYPE),pointer TYPES::NONLINEAR\_- TRUSTREGION\_SOLVER\_TYPE::NONLINEAR\_SOLVER

A pointer to the nonlinear solver.

Definition at line 1418 of file types.f90.

#### 7.287.2.2 TYPE(PETSC\_SNES\_TYPE) TYPES::NONLINEAR\_TRUSTREGION\_SOLVER\_- TYPE::SNES

The PETSc nonlinear solver object.

Definition at line 1422 of file types.f90.

**7.287.2.3 INTEGER(INTG) TYPES::NONLINEAR\_TRUSTREGION\_SOLVER\_-  
TYPE::SOLVER\_LIBRARY**

The library type for the nonlinear solver.

**See also:**

[SOLVER\\_ROUTINES::SolverLibraries](#), [SOLVER\\_ROUTINES](#)

Definition at line 1419 of file types.f90.

**7.287.2.4 REAL(DP) TYPES::NONLINEAR\_TRUSTREGION\_SOLVER\_-  
TYPE::TRUSTREGION\_DELTA0**

The trust region delta0.

Definition at line 1421 of file types.f90.

**7.287.2.5 REAL(DP) TYPES::NONLINEAR\_TRUSTREGION\_SOLVER\_-  
TYPE::TRUSTREGION\_TOLERANCE**

The trust region tolerance.

Definition at line 1420 of file types.f90.

## 7.288 TYPES::PROBLEM\_CONTROL\_TYPE Struct Reference

Collaboration diagram for TYPES::PROBLEM\_CONTROL\_TYPE:

### Public Attributes

- TYPE([PROBLEM\\_TYPE](#)), pointer [PROBLEM](#)  
*A pointer back to the problem.*
- LOGICAL [CONTROL\\_FINISHED](#)  
*Is .TRUE. if the problem control has finished being created, .FALSE. if not.*
- INTEGER(INTG) [CONTROL\\_TYPE](#)

### 7.288.1 Detailed Description

Definition at line 1674 of file types.f90.

### 7.288.2 Member Data Documentation

#### 7.288.2.1 LOGICAL TYPES::PROBLEM\_CONTROL\_TYPE::CONTROL\_FINISHED

Is .TRUE. if the problem control has finished being created, .FALSE. if not.

Definition at line 1676 of file types.f90.

#### 7.288.2.2 INTEGER(INTG) TYPES::PROBLEM\_CONTROL\_TYPE::CONTROL\_TYPE

Definition at line 1677 of file types.f90.

#### 7.288.2.3 TYPE(PROBLEM\_TYPE),pointer TYPES::PROBLEM\_CONTROL\_TYPE::PROBLEM

A pointer back to the problem.

Definition at line 1675 of file types.f90.

## 7.289 TYPES::PROBLEM\_LINEAR\_DATA\_TYPE Struct Reference

Contains information on any data required for a linear solution.

Collaboration diagram for TYPES::PROBLEM\_LINEAR\_DATA\_TYPE:

### Public Attributes

- TYPE(SOLUTION\_TYPE), pointer SOLUTION

*A pointer to the problem solution.*

#### 7.289.1 Detailed Description

Contains information on any data required for a linear solution.

Definition at line 1659 of file types.f90.

#### 7.289.2 Member Data Documentation

##### 7.289.2.1 TYPE(SOLUTION\_TYPE),pointer TYPES::PROBLEM\_LINEAR\_DATA\_TYPE::SOLUTION

A pointer to the problem solution.

Definition at line 1660 of file types.f90.

## 7.290 TYPES::PROBLEM\_NONLINEAR\_DATA\_TYPE Struct Reference

Contains information on any data required for a non-linear solution.

Collaboration diagram for TYPES::PROBLEM\_NONLINEAR\_DATA\_TYPE:

### Public Attributes

- TYPE(SOLUTION\_TYPE), pointer SOLUTION  
*A pointer to the problem solution.*
- INTEGER(INTG) NUMBER\_OF\_ITERATIONS

#### 7.290.1 Detailed Description

Contains information on any data required for a non-linear solution.

Definition at line 1664 of file types.f90.

#### 7.290.2 Member Data Documentation

##### 7.290.2.1 INTEGER(INTG) TYPES::PROBLEM\_NONLINEAR\_DATA\_TYPE::NUMBER\_OF\_ITERATIONS

Definition at line 1666 of file types.f90.

##### 7.290.2.2 TYPE(SOLUTION\_TYPE),pointer TYPES::PROBLEM\_NONLINEAR\_DATA\_TYPE::SOLUTION

A pointer to the problem solution.

Definition at line 1665 of file types.f90.

## 7.291 TYPES::PROBLEM\_PTR\_TYPE Struct Reference

A buffer type to allow for an array of pointers to a PROBLEM\_TYPE.

Collaboration diagram for TYPES::PROBLEM\_PTR\_TYPE:

### Public Attributes

- TYPE([PROBLEM\\_TYPE](#)), pointer PTR  
*The pointer to the problem.*

#### 7.291.1 Detailed Description

A buffer type to allow for an array of pointers to a PROBLEM\_TYPE.

See also:

[TYPES:PROBLEM\\_TYPE](#)

Definition at line 1698 of file types.f90.

#### 7.291.2 Member Data Documentation

##### 7.291.2.1 TYPE(PROBLEM\_TYPE),pointer TYPES::PROBLEM\_PTR\_TYPE::PTR

The pointer to the problem.

Definition at line 1699 of file types.f90.

## 7.292 TYPES::PROBLEM\_TIME\_DATA\_TYPE Struct Reference

Contains information on any data required for a time-dependent solution.

Collaboration diagram for TYPES::PROBLEM\_TIME\_DATA\_TYPE:

### Public Attributes

- TYPE(SOLUTION\_TYPE), pointer SOLUTION  
*A pointer to the problem solution.*

#### 7.292.1 Detailed Description

Contains information on any data required for a time-dependent solution.

Definition at line 1670 of file types.f90.

#### 7.292.2 Member Data Documentation

##### 7.292.2.1 TYPE(SOLUTION\_TYPE),pointer TYPES::PROBLEM\_TIME\_DATA\_- TYPE::SOLUTION

A pointer to the problem solution.

Definition at line 1671 of file types.f90.

## 7.293 TYPES::PROBLEM\_TYPE Struct Reference

Contains information for a problem.

Collaboration diagram for TYPES::PROBLEM\_TYPE:

### Public Attributes

- INTEGER(INTG) **USER\_NUMBER**  
*The user defined identifier for the problem. The user number must be unique.*
- INTEGER(INTG) **GLOBAL\_NUMBER**  
*The global number of the problem in the list of problems.*
- LOGICAL **PROBLEM\_FINISHED**  
*Is .TRUE. if the problem has finished being created, .FALSE. if not.*
- TYPE(PROBLEMS\_TYPE), pointer **PROBLEMS**  
*A pointer to the problems for this problem.*
- INTEGER(INTG) **CLASS**  
*The problem specification class identifier.*
- INTEGER(INTG) **TYPE**  
*The problem specification type identifier.*
- INTEGER(INTG) **SUBTYPE**  
*The problem specification subtype identifier.*
- TYPE(PROBLEM\_CONTROL\_TYPE), pointer **CONTROL**  
*A pointer to the control information for the problem.*
- INTEGER(INTG) **NUMBER\_OF\_SOLUTIONS**  
*The number of solutions in the problem.*
- TYPE(SOLUTION\_PTR\_TYPE), allocatable **SOLUTIONS**  
*A pointer to the solution information for the problem.*

### 7.293.1 Detailed Description

Contains information for a problem.

Definition at line 1681 of file types.f90.

### 7.293.2 Member Data Documentation

#### 7.293.2.1 INTEGER(INTG) TYPES::PROBLEM\_TYPE::CLASS

The problem specification class identifier.

Definition at line 1687 of file types.f90.

**7.293.2.2 TYPE (PROBLEM\_CONTROL\_TYPE),pointer TYPES::PROBLEM\_TYPE::CONTROL**

A pointer to the control information for the problem.

Definition at line 1691 of file types.f90.

**7.293.2.3 INTEGER(INTG) TYPES::PROBLEM\_TYPE::GLOBAL\_NUMBER**

The global number of the problem in the list of problems.

Definition at line 1683 of file types.f90.

**7.293.2.4 INTEGER(INTG) TYPES::PROBLEM\_TYPE::NUMBER\_OF\_SOLUTIONS**

The number of solutions in the problem.

Definition at line 1693 of file types.f90.

**7.293.2.5 LOGICAL TYPES::PROBLEM\_TYPE::PROBLEM\_FINISHED**

Is .TRUE. if the problem has finished being created, .FALSE. if not.

Definition at line 1684 of file types.f90.

**7.293.2.6 TYPE(PROBLEMS\_TYPE),pointer TYPES::PROBLEM\_TYPE::PROBLEMS**

A pointer to the problems for this problem.

Definition at line 1685 of file types.f90.

**7.293.2.7 TYPE(SOLUTION\_PTR\_TYPE),allocatable TYPES::PROBLEM\_TYPE::SOLUTIONS**

A pointer to the solution information for the problem.

Definition at line 1694 of file types.f90.

**7.293.2.8 INTEGER(INTG) TYPES::PROBLEM\_TYPE::SUBTYPE**

The problem specification subtype identifier.

Definition at line 1689 of file types.f90.

**7.293.2.9 INTEGER(INTG) TYPES::PROBLEM\_TYPE::TYPE**

The problem specification type identifier.

Definition at line 1688 of file types.f90.

**7.293.2.10 INTEGER(INTG) TYPES::PROBLEM\_TYPE::USER\_NUMBER**

The user defined identifier for the problem. The user number must be unique.

Definition at line 1682 of file types.f90.

## 7.294 TYPES::PROBLEMS\_TYPE Struct Reference

Contains information on the problems defined on a region.

Collaboration diagram for TYPES::PROBLEMS\_TYPE:

### Public Attributes

- INTEGER(INTG) [NUMBER\\_OF\\_PROBLEMS](#)  
*The number of problems defined.*
- TYPE([PROBLEM\\_PTR\\_TYPE](#)), pointer [PROBLEMS](#)  
*The array of pointers to the problems.*

### 7.294.1 Detailed Description

Contains information on the problems defined on a region.

Definition at line 1703 of file types.f90.

### 7.294.2 Member Data Documentation

#### 7.294.2.1 INTEGER(INTG) TYPES::PROBLEMS\_TYPE::NUMBER\_OF\_PROBLEMS

The number of problems defined.

Definition at line 1704 of file types.f90.

#### 7.294.2.2 TYPE(PROBLEM\_PTR\_TYPE),pointer TYPES::PROBLEMS\_TYPE::PROBLEMS

The array of pointers to the problems.

Definition at line 1705 of file types.f90.

## 7.295 TYPES::QUADRATURE\_SCHEME\_PTR\_TYPE Struct Reference

A buffer type to allow for an array of pointers to a QUADRATURE\_SCHEME\_TYPE.

Collaboration diagram for TYPES::QUADRATURE\_SCHEME\_PTR\_TYPE:

### Public Attributes

- TYPE(QUADRATURE\_SCHEME\_TYPE), pointer PTR  
*A pointer to the quadrature scheme.*

#### 7.295.1 Detailed Description

A buffer type to allow for an array of pointers to a QUADRATURE\_SCHEME\_TYPE.

See also:

[TYPES::QUADRATURE\\_SCHEME\\_TYPE](#)

Definition at line 96 of file types.f90.

#### 7.295.2 Member Data Documentation

##### 7.295.2.1 TYPE(QUADRATURE\_SCHEME\_TYPE),pointer TYPES::QUADRATURE\_SCHEME\_PTR\_TYPE::PTR

A pointer to the quadrature scheme.

Definition at line 97 of file types.f90.

## 7.296 TYPES::QUADRATURE\_SCHEME\_TYPE Struct Reference

Contains information for a particular quadrature scheme.

Collaboration diagram for TYPES::QUADRATURE\_SCHEME\_TYPE:

### Public Attributes

- INTEGER(INTG) [GLOBAL\\_NUMBER](#)

*The global number of the quadrature scheme in the list of quadrature schemes for a particular quadrature.*

- TYPE([QUADRATURE\\_TYPE](#)), pointer [QUADRATURE](#)

*The pointer back to the quadrature for a particular quadrature scheme.*

- INTEGER(INTG) [NUMBER\\_OF\\_GAUSS](#)

*The number of gauss points for the quadrature scheme.*

- REAL(DP), allocatable [GAUSS\\_POSITIONS](#)

*GAUSS\_POSITIONS(nic,ng). The positions in the nic'th xi coordinate of Gauss point ng. Old [CMISS](#) name XIG(ni,ng,nb).*

- REAL(DP), allocatable [GAUSS\\_WEIGHTS](#)

*GAUSS\_WEIGHTS(ng). The weight applied to Gauss point ng. Old [CMISS](#) name WG(ng,nb).*

- REAL(DP), allocatable [GAUSS\\_BASIS\\_FNS](#)

*GAUSS\_BASIS\_FNS(ns,nu,ng). The value of the basis functions evaluated at Gauss point ng for the nu'th derivative of the basis function associated with the ns'th element parameter. Old [CMISS](#) name PG(ns,nu,ng,nb).*

### 7.296.1 Detailed Description

Contains information for a particular quadrature scheme.

#### [Todo](#)

Also evaluate the product of the basis functions at gauss points for speed???

Definition at line 86 of file types.f90.

### 7.296.2 Member Data Documentation

#### 7.296.2.1 REAL(DP),allocatable TYPES::QUADRATURE\_SCHEME\_TYPE::GAUSS\_BASIS\_FNS

GAUSS\_BASIS\_FNS(ns,nu,ng). The value of the basis functions evaluated at Gauss point ng for the nu'th derivative of the basis function associated with the ns'th element parameter. Old [CMISS](#) name PG(ns,nu,ng,nb).

Definition at line 92 of file types.f90.

**7.296.2.2 REAL(DP),allocatable TYPES::QUADRATURE\_SCHEME\_TYPE::GAUSS\_POSITIONS**

GAUSS\_POSITIONS(nic,ng). The positions in the nic'th xi coordinate of Gauss point ng. Old [CMISS](#) name XIG(ni,ng,nb).

Definition at line 90 of file types.f90.

**7.296.2.3 REAL(DP),allocatable TYPES::QUADRATURE\_SCHEME\_TYPE::GAUSS\_WEIGHTS**

GAUSS\_WEIGHTS(ng). The weight applied to Gauss point ng. Old [CMISS](#) name WG(ng,nb).

Definition at line 91 of file types.f90.

**7.296.2.4 INTEGER(INTG) TYPES::QUADRATURE\_SCHEME\_TYPE::GLOBAL\_NUMBER**

The global number of the quadrature scheme in the list of quadrature schemes for a particular quadrature.

Definition at line 87 of file types.f90.

**7.296.2.5 INTEGER(INTG) TYPES::QUADRATURE\_SCHEME\_TYPE::NUMBER\_OF\_GAUSS**

The number of gauss points for the quadrature scheme.

Definition at line 89 of file types.f90.

**7.296.2.6 TYPE(QUADRATURE\_TYPE),pointer TYPES::QUADRATURE\_SCHEME\_TYPE::QUADRATURE**

The pointer back to the quadrature for a particular quadrature scheme.

Definition at line 88 of file types.f90.

## 7.297 TYPES::QUADRATURE\_TYPE Struct Reference

Contains information on the quadrature to be used for integrating a basis.

Collaboration diagram for TYPES::QUADRATURE\_TYPE:

### Public Attributes

- INTEGER(INTG) **TYPE**

*The type of the quadrature.*

- TYPE(BASIS\_TYPE), pointer **BASIS**

*The pointer back to the basis.*

- INTEGER(INTG), allocatable **NUMBER\_OF\_GAUSS\_XI**

*NUMBER\_OF\_GAUSS\_XI( $ni$ ). For standard Gauss schemes the number of Gauss points to be used in the  $ni$ 'th  $xi$  direction.*

- INTEGER(INTG) **GAUSS\_ORDER**

*For simplex Gauss schemes the order of the Quadrature scheme i.e., the order/dimension of the polynomial that can be integrated.*

- TYPE(QUADRATURE\_SCHEME\_PTR\_TYPE), allocatable **QUADRATURE\_SCHEME\_MAP**

*QUADRATURE\_SCHEME\_MAP(scheme\_idx). The pointer map to the defined quadrature schemes. The size of array is given by BASIS\_ROUTINES::BASIS\_NUMBER\_OF\_QUADRATURE\_SCHEME\_TYPES. If the quadrature scheme is not defined for the particular type then the array element is NULL.*

- INTEGER(INTG) **NUMBER\_OF\_SCHEMES**

*The number of quadrature schemes defined for this quadrature.*

- TYPE(QUADRATURE\_SCHEME\_PTR\_TYPE), pointer **SCHEMES**

*SCHEMES(scheme\_idx). The array of pointers to the quadrature schemes defined for the basis. scheme\_idx must be between 1 and QUADRATURE\_TYPE::NUMBER\_OF\_SCHEMES.*

### 7.297.1 Detailed Description

Contains information on the quadrature to be used for integrating a basis.

Definition at line 101 of file types.f90.

### 7.297.2 Member Data Documentation

#### 7.297.2.1 TYPE(BASIS\_TYPE),pointer TYPES::QUADRATURE\_TYPE::BASIS

The pointer back to the basis.

Definition at line 103 of file types.f90.

### **7.297.2.2 INTEGER(INTG) TYPES::QUADRATURE\_TYPE::GAUSS\_ORDER**

For simplex Gauss schemes the order of the Quadrature scheme i.e., the order/dimension of the polynomial that can be integrated.

Definition at line 105 of file types.f90.

### **7.297.2.3 INTEGER(INTG),allocatable TYPES::QUADRATURE\_TYPE::NUMBER\_OF\_GAUSS\_XI**

NUMBER\_OF\_GAUSS\_XI(ni). For standard Gauss schemes the number of Gauss points to be used in the ni'th xi direction.

Definition at line 104 of file types.f90.

### **7.297.2.4 INTEGER(INTG) TYPES::QUADRATURE\_TYPE::NUMBER\_OF\_SCHEMES**

The number of quadrature schemes defined for this quadrature.

Definition at line 107 of file types.f90.

### **7.297.2.5 TYPE(QUADRATURE\_SCHEME\_PTR\_TYPE),allocatable TYPES::QUADRATURE\_TYPE::QUADRATURE\_SCHEME\_MAP**

QUADRATURE\_SCHEME\_MAP(scheme\_idx). The pointer map to the defined quadrature schemes. The size of array is given by BASIS\_ROUTINES::BASIS\_NUMBER\_OF\_QUADRATURE\_SCHEME\_TYPES. If the quadrature scheme is not defined for the particular type then the array element is NULL.

**See also:**

BASIS\_ROUTINES\_QuadratureSchemes.

Definition at line 106 of file types.f90.

### **7.297.2.6 TYPE(QUADRATURE\_SCHEME\_PTR\_TYPE),pointer TYPES::QUADRATURE\_TYPE::SCHEMES**

SCHEMES(scheme\_idx). The array of pointers to the quadrature schemes defined for the basis. scheme\_idx must be between 1 and QUADRATURE\_TYPE::NUMBER\_OF\_SCHEMES.

Definition at line 108 of file types.f90.

### **7.297.2.7 INTEGER(INTG) TYPES::QUADRATURE\_TYPE::TYPE**

The type of the quadrature.

**See also:**

BASIS\_ROUTINES\_QuadratureTypes

Definition at line 102 of file types.f90.

## 7.298 TYPES::REGION\_PTR\_TYPE Struct Reference

A buffer type to allow for an array of pointers to a REGION\_TYPE.

Collaboration diagram for TYPES::REGION\_PTR\_TYPE:

### Public Attributes

- TYPE(REGION\_TYPE), pointer PTR

*The pointer to the region.*

### 7.298.1 Detailed Description

A buffer type to allow for an array of pointers to a REGION\_TYPE.

Definition at line 1714 of file types.f90.

### 7.298.2 Member Data Documentation

#### 7.298.2.1 TYPE(REGION\_TYPE),pointer TYPES::REGION\_PTR\_TYPE::PTR

The pointer to the region.

Definition at line 1715 of file types.f90.

## 7.299 TYPES::REGION\_TYPE Struct Reference

Contains information for a region.

Collaboration diagram for TYPES::REGION\_TYPE:

### Public Attributes

- INTEGER(INTG) **USER\_NUMBER**

*The user defined identifier for the region. The user number must be unique.*

- LOGICAL **REGION\_FINISHED**

*Is .TRUE. if the region has finished being created, .FALSE. if not.*

- TYPE(**VARYING\_STRING**) **LABEL**

*A user defined label for the region.*

- TYPE(**COORDINATE\_SYSTEM\_TYPE**), pointer **COORDINATE\_SYSTEM**

*A pointer to the coordinate system used by the region.*

- TYPE(**NODES\_TYPE**), pointer **NODES**

*A pointer to the nodes defined on the region.*

- TYPE(**MESHES\_TYPE**), pointer **MESHES**

*A pointer to the meshes defined on the region.*

- TYPE(**FIELDS\_TYPE**), pointer **FIELDS**

*A pointer to the fields defined on the region.*

- TYPE(**EQUATIONS\_SETS\_TYPE**), pointer **EQUATIONS\_SETS**

*A pointer to the equation sets defined on the region.*

- TYPE(**REGION\_TYPE**), pointer **PARENT\_REGION**

*A pointer to the parent region for the region. If the region has no parent region then it is the global (world) region and PARENT\_REGION is NULL.*

- INTEGER(INTG) **NUMBER\_OF\_SUB\_REGIONS**

*The number of sub-regions defined for the region.*

- TYPE(**REGION\_PTR\_TYPE**), pointer **SUB\_REGIONS**

*An array of pointers to the sub-regions defined on the region.*

### 7.299.1 Detailed Description

Contains information for a region.

Definition at line 1719 of file types.f90.

## 7.299.2 Member Data Documentation

### 7.299.2.1 TYPE(COORDINATE\_SYSTEM\_TYPE),pointer TYPES::REGION\_TYPE::COORDINATE\_SYSTEM

A pointer to the coordinate system used by the region.

Definition at line 1723 of file types.f90.

### 7.299.2.2 TYPE(EQUATIONS\_SETS\_TYPE),pointer TYPES::REGION\_TYPE::EQUATIONS\_SETS

A pointer to the equation sets defined on the region.

Definition at line 1727 of file types.f90.

### 7.299.2.3 TYPE(FIELDS\_TYPE),pointer TYPES::REGION\_TYPE::FIELDS

A pointer to the fields defined on the region.

Definition at line 1726 of file types.f90.

### 7.299.2.4 TYPE(VARYING\_STRING) TYPES::REGION\_TYPE::LABEL

A user defined label for the region.

Definition at line 1722 of file types.f90.

### 7.299.2.5 TYPE(MESHES\_TYPE),pointer TYPES::REGION\_TYPE::MESHES

A pointer to the meshes defined on the region.

Definition at line 1725 of file types.f90.

### 7.299.2.6 TYPE(NODES\_TYPE),pointer TYPES::REGION\_TYPE::NODES

A pointer to the nodes defined on the region.

Definition at line 1724 of file types.f90.

### 7.299.2.7 INTEGER(INTG) TYPES::REGION\_TYPE::NUMBER\_OF\_SUB\_REGIONS

The number of sub-regions defined for the region.

Definition at line 1729 of file types.f90.

### 7.299.2.8 TYPE(REGION\_TYPE),pointer TYPES::REGION\_TYPE::PARENT\_REGION

A pointer to the parent region for the region. If the region has no parent region then it is the global (world) region and PARENT\_REGION is NULL.

Definition at line 1728 of file types.f90.

**7.299.2.9 LOGICAL TYPES::REGION\_TYPE::REGION\_FINISHED**

Is .TRUE. if the region has finished being created, .FALSE. if not.

Definition at line 1721 of file types.f90.

**7.299.2.10 TYPE(REGION\_PTR\_TYPE),pointer TYPES::REGION\_TYPE::SUB\_REGIONS**

An array of pointers to the sub-regions defined on the region.

Definition at line 1730 of file types.f90.

**7.299.2.11 INTEGER(INTG) TYPES::REGION\_TYPE::USER\_NUMBER**

The user defined identifier for the region. The user number must be unique.

Definition at line 1720 of file types.f90.

## **7.300 TYPES::SOLUTION\_MAPPING\_CREATE\_VALUES\_- CACHE\_TYPE Struct Reference**

Contains information about the cached create values for a solution mapping.

### **Public Attributes**

- INTEGER(INTG), allocatable [MATRIX\\_VARIABLE\\_TYPES](#)

*MATRIX\_VARIABLE\_TYPES(0:..,equations\_set\_idx,matrix\_idx)*. The list of matrix variable types in the equations\_set\_idx'th equations set for the matrix\_idx'th solver matrix. *MATRIX\_VARIABLE\_TYPES(0,equations\_set\_idx,matrix\_idx)* is the number of variable types in the equations\_set\_idx'th equations set mapped to the matrix\_idx'th solver matrix and *MATRIX\_VARIABLE\_TYPES(1..,equations\_set\_idx,matrix\_idx)* is the list of the variable types in the equations set.

- INTEGER, allocatable [RESIDUAL\\_VARIABLE\\_TYPE](#)

*RESIDUAL\_VARIABLE\_TYPE(equations\_set\_idx)*. The variable type that is mapped to the solution residual for the equations\_set\_idx'th equations set.

- INTEGER, allocatable [RHS\\_VARIABLE\\_TYPE](#)

*RHS\_VARIABLE\_TYPE(equations\_set\_idx)*. The variable type that is mapped to the solution RHS for the equations\_set\_idx'th equations set.

- INTEGER, allocatable [SOURCE\\_VARIABLE\\_TYPE](#)

*SOURCE\_VARIABLE\_TYPE(equations\_set\_idx)*. The source variable type that is mapped to the RHS for the equations\_set\_idx'th equations set.

### **7.300.1 Detailed Description**

Contains information about the cached create values for a solution mapping.

Definition at line 1603 of file types.f90.

### **7.300.2 Member Data Documentation**

#### **7.300.2.1 INTEGER(INTG),allocatable TYPES::SOLUTION\_MAPPING\_CREATE\_VALUES\_- CACHE\_TYPE::MATRIX\_VARIABLE\_TYPES**

*MATRIX\_VARIABLE\_TYPES(0:..,equations\_set\_idx,matrix\_idx)*. The list of matrix variable types in the equations\_set\_idx'th equations set for the matrix\_idx'th solver matrix. *MATRIX\_VARIABLE\_TYPES(0,equations\_set\_idx,matrix\_idx)* is the number of variable types in the equations\_set\_idx'th equations set mapped to the matrix\_idx'th solver matrix and *MATRIX\_VARIABLE\_TYPES(1..,equations\_set\_idx,matrix\_idx)* is the list of the variable types in the equations set.

Definition at line 1604 of file types.f90.

#### **7.300.2.2 INTEGER,allocatable TYPES::SOLUTION\_MAPPING\_CREATE\_VALUES\_- CACHE\_TYPE::RESIDUAL\_VARIABLE\_TYPE**

*RESIDUAL\_VARIABLE\_TYPE(equations\_set\_idx)*. The variable type that is mapped to the solution residual for the equations\_set\_idx'th equations set.

Definition at line 1605 of file types.f90.

**7.300.2.3 INTEGER,allocatable TYPES::SOLUTION\_MAPPING\_CREATE\_VALUES\_-  
CACHE\_TYPE::RHS\_VARIABLE\_TYPE**

RHS\_VARIABLE\_TYPE(equations\_set\_idx). The variable type that is mapped to the solution RHS for the equations\_set\_idx'th equations set.

Definition at line 1606 of file types.f90.

**7.300.2.4 INTEGER,allocatable TYPES::SOLUTION\_MAPPING\_CREATE\_VALUES\_-  
CACHE\_TYPE::SOURCE\_VARIABLE\_TYPE**

SOURCE\_VARIABLE\_TYPE(equations\_set\_idx). The source variable type that is mapped to the RHS for the equations\_set\_idx'th equations set.

Definition at line 1607 of file types.f90.

## 7.301 TYPES::SOLUTION\_MAPPING\_TYPE Struct Reference

Contains information on the solution mapping between the global equation sets and the solver matrices.

Collaboration diagram for TYPES::SOLUTION\_MAPPING\_TYPE:

### Public Attributes

- TYPE(SOLUTION\_TYPE), pointer SOLUTION  
*A pointer to the solution for this mapping.*
- LOGICAL SOLUTION\_MAPPING\_FINISHED  
*Is .TRUE. if the solution mapping has finished being created, .FALSE. if not.*
- INTEGER(INTG) NUMBER\_OF\_ROWS  
*The number of (local) rows in the solver matrices.*
- INTEGER(INTG) NUMBER\_OF\_GLOBAL\_ROWS  
*The number of global rows in the solver matrices.*
- INTEGER(INTG) NUMBER\_OF\_SOLVER\_MATRICES  
*The number of solution matrices in this mapping.*
- INTEGER(INTG) NUMBER\_OF\_EQUATIONS\_SETS  
*The number of equations sets in the solution mapping.*
- TYPE(EQUATIONS\_SET\_PTR\_TYPE), allocatable EQUATIONS\_SETS  
*The list of equations sets that are in this solution mapping.*
- TYPE(EQUATIONS\_SET\_TO\_SOLVER\_MAP\_TYPE), allocatable EQUATIONS\_SET\_TO\_-  
**SOLVER\_MAP**  
*EQUATIONS\_SET\_TO\_SOLVER\_MAP(equations\_set\_idx). The mapping from the equations\_set\_idx'th equations set to the solver matrices.*
- TYPE(SOLVER\_COL\_TO\_EQUATIONS\_SETS\_MAP\_TYPE), allocatable SOLVER\_COL\_TO\_-  
**EQUATIONS\_SETS\_MAP**  
*SOLVER\_TO\_EQUATIONS\_SET\_MAP(solver\_matrix\_idx). The mapping from the solver\_matrix\_idx'th solver matrix to the equations set.*
- TYPE(SOLVER\_ROW\_TO\_EQUATIONS\_SET\_MAP\_TYPE), allocatable SOLVER\_ROW\_TO\_-  
**EQUATIONS\_SET\_MAPS**  
*SOLVER\_ROW\_TO\_EQUATIONS\_SET\_MAPS(row\_idx). The mappings from the row\_idx'th solver row to the equations set rows.*
- LOGICAL HAVE\_JACOBIAN  
*Is .TRUE. if the Jacobian exists for nonlinear problems.*
- TYPE(DOMAIN\_MAPPING\_TYPE), pointer ROW\_DOFS\_MAPPING  
*The domain mapping for the solver rows.*

- **TYPE(SOLUTION\_MAPPING\_CREATE\_VALUES\_CACHE\_TYPE), pointer** **CREATE\_VALUES\_CACHE**

*The create values cache for the solution mapping.*

### 7.301.1 Detailed Description

Contains information on the solution mapping between the global equation sets and the solver matrices.

Definition at line 1611 of file types.f90.

### 7.301.2 Member Data Documentation

#### 7.301.2.1 **TYPE(SOLUTION\_MAPPING\_CREATE\_VALUES\_CACHE\_TYPE),pointer** **TYPES::SOLUTION\_MAPPING\_TYPE::CREATE\_VALUES\_CACHE**

The create values cache for the solution mapping.

Definition at line 1624 of file types.f90.

#### 7.301.2.2 **TYPE(EQUATIONS\_SET\_TO\_SOLVER\_MAP\_TYPE),allocatable** **TYPES::SOLUTION\_MAPPING\_TYPE::EQUATIONS\_SET\_TO\_SOLVER\_MAP**

EQUATIONS\_SET\_TO\_SOLVER\_MAP(equations\_set\_idx). The mapping from the equations\_set\_idx'the equations set to the solver matrices.

Definition at line 1619 of file types.f90.

#### 7.301.2.3 **TYPE(EQUATIONS\_SET\_PTR\_TYPE),allocatable** **TYPES::SOLUTION\_MAPPING\_TYPE::EQUATIONS\_SETS**

The list of equations sets that are in this solution mapping.

Definition at line 1618 of file types.f90.

#### 7.301.2.4 **LOGICAL TYPES::SOLUTION\_MAPPING\_TYPE::HAVE\_JACOBIAN**

Is .TRUE. if the Jacobian exists for nonlinear problems.

Definition at line 1622 of file types.f90.

#### 7.301.2.5 **INTEGER(INTG) TYPES::SOLUTION\_MAPPING\_TYPE::NUMBER\_OF\_EQUATIONS\_SETS**

The number of equations sets in the solution mapping.

Definition at line 1617 of file types.f90.

#### 7.301.2.6 **INTEGER(INTG) TYPES::SOLUTION\_MAPPING\_TYPE::NUMBER\_OF\_GLOBAL\_ROWS**

The number of global rows in the solver matrices.

Definition at line 1615 of file types.f90.

#### 7.301.2.7 INTEGER(INTG) TYPES::SOLUTION\_MAPPING\_TYPE::NUMBER\_OF\_ROWS

The number of (local) rows in the solver matrices.

Definition at line 1614 of file types.f90.

#### 7.301.2.8 INTEGER(INTG) TYPES::SOLUTION\_MAPPING\_TYPE::NUMBER\_OF\_- SOLVER\_MATRICES

The number of solution matrices in this mapping.

Definition at line 1616 of file types.f90.

#### 7.301.2.9 TYPE(DOMAIN\_MAPPING\_TYPE),pointer TYPES::SOLUTION\_MAPPING\_- TYPE::ROW\_DOFS\_MAPPING

The domain mapping for the solver rows.

Definition at line 1623 of file types.f90.

#### 7.301.2.10 TYPE(SOLUTION\_TYPE),pointer TYPES::SOLUTION\_MAPPING\_- TYPE::SOLUTION

A pointer to the solution for this mapping.

Definition at line 1612 of file types.f90.

#### 7.301.2.11 LOGICAL TYPES::SOLUTION\_MAPPING\_TYPE::SOLUTION\_MAPPING\_- FINISHED

Is .TRUE. if the solution mapping has finished being created, .FALSE. if not.

Definition at line 1613 of file types.f90.

#### 7.301.2.12 TYPE(SOLVER\_COL\_TO\_EQUATIONS\_SETS\_MAP\_TYPE),allocatable TYPES::SOLUTION\_MAPPING\_TYPE::SOLVER\_COL\_TO\_EQUATIONS\_SETS\_- MAP

SOLVER\_TO\_EQUATIONS\_SET\_MAP(solver\_matrix\_idx). The mapping from the solver\_matrix\_-idx'th solver matrix to the equations set.

Definition at line 1620 of file types.f90.

#### 7.301.2.13 TYPE(SOLVER\_ROW\_TO\_EQUATIONS\_SET\_MAP\_TYPE),allocatable TYPES::SOLUTION\_MAPPING\_TYPE::SOLVER\_ROW\_TO\_EQUATIONS\_SET\_- MAPS

SOLVER\_ROW\_TO\_EQUATIONS\_SET\_MAPS(row\_idx). The mappings from the row\_idx'th solver row to the equations set rows.

Definition at line 1621 of file types.f90.

## 7.302 TYPES::SOLUTION\_PTR\_TYPE Struct Reference

Collaboration diagram for TYPES::SOLUTION\_PTR\_TYPE:

### Public Attributes

- TYPE(SOLUTION\_TYPE), pointer PTR

#### 7.302.1 Detailed Description

Definition at line 1648 of file types.f90.

#### 7.302.2 Member Data Documentation

##### 7.302.2.1 TYPE(SOLUTION\_TYPE),pointer TYPES::SOLUTION\_PTR\_TYPE::PTR

Definition at line 1649 of file types.f90.

## 7.303 TYPES::SOLUTION\_TYPE Struct Reference

Contains information on the solution of a problem.

Collaboration diagram for TYPES::SOLUTION\_TYPE:

### Public Attributes

- INTEGER(INTG) [SOLUTION\\_NUMBER](#)  
*The number of the solution.*
- TYPE([PROBLEM\\_TYPE](#)), pointer [PROBLEM](#)  
*A pointer back to the problem for this solution.*
- LOGICAL [SOLUTION\\_FINISHED](#)  
*Is .TRUE. if the problem solution has finished being created, .FALSE. if not.*
- INTEGER(INTG) [LINEARITY](#)
- TYPE([EQUATIONS\\_SET\\_TYPE](#)), pointer [EQUATIONS\\_SET\\_TO\\_ADD](#)  
*The next equations set to add to the solution.*
- INTEGER(INTG) [EQUATIONS\\_SET\\_ADDED\\_INDEX](#)  
*The index of the last successfully added equations set.*
- TYPE([SOLUTION\\_MAPPING\\_TYPE](#)), pointer [SOLUTION\\_MAPPING](#)  
*A pointer to the solution mapping for the solution.*
- TYPE([SOLVER\\_TYPE](#)), pointer [SOLVER](#)  
*A pointer to the solver for the problem.*

### 7.303.1 Detailed Description

Contains information on the solution of a problem.

Definition at line 1634 of file types.f90.

### 7.303.2 Member Data Documentation

#### 7.303.2.1 INTEGER(INTG) TYPES::SOLUTION\_TYPE::EQUATIONS\_SET\_ADDED\_INDEX

The index of the last successfully added equations set.

Definition at line 1643 of file types.f90.

#### 7.303.2.2 TYPE(EQUATIONS\_SET\_TYPE),pointer TYPES::SOLUTION\_TYPE::EQUATIONS\_SET\_TO\_ADD

The next equations set to add to the solution.

Definition at line 1642 of file types.f90.

**7.303.2.3 INTEGER(INTG) TYPES::SOLUTION\_TYPE::LINEARITY**

Definition at line 1638 of file types.f90.

**7.303.2.4 TYPE(PROBLEM\_TYPE),pointer TYPES::SOLUTION\_TYPE::PROBLEM**

A pointer back to the problem for this solution.

Definition at line 1636 of file types.f90.

**7.303.2.5 LOGICAL TYPES::SOLUTION\_TYPE::SOLUTION\_FINISHED**

Is .TRUE. if the problem solution has finished being created, .FALSE. if not.

Definition at line 1637 of file types.f90.

**7.303.2.6 TYPE(SOLUTION\_MAPPING\_TYPE),pointer TYPES::SOLUTION\_-  
TYPE::SOLUTION\_MAPPING**

A pointer to the solution mapping for the solution.

Definition at line 1644 of file types.f90.

**7.303.2.7 INTEGER(INTG) TYPES::SOLUTION\_TYPE::SOLUTION\_NUMBER**

The number of the solution.

Definition at line 1635 of file types.f90.

**7.303.2.8 TYPE(SOLVER\_TYPE),pointer TYPES::SOLUTION\_TYPE::SOLVER**

A pointer to the solver for the problem.

Definition at line 1645 of file types.f90.

## 7.304 TYPES::SOLVER\_COL\_TO\_EQUATIONS\_MAP\_TYPE Struct Reference

Contains information about the mapping from a solver matrix column to equations matrices and variables.

### Public Attributes

- INTEGER(INTG) [NUMBER\\_OF\\_LINEAR\\_EQUATIONS\\_MATRICES](#)  
*The number of linear equations matrices the solver column is mapped to in this equations set.*
- INTEGER(INTG), allocatable [EQUATIONS\\_MATRIX\\_NUMBERS](#)  
*EQUATIONS\_MATRIX\_NUMBERS(i). The i'th equations matrix number in the equations that the solver column is mapped to.*
- INTEGER(INTG), allocatable [EQUATIONS\\_COL\\_NUMBERS](#)  
*EQUATIONS\_COL\_NUMBERS(i). The i'th equations column number in the equation set the solver column is mapped to.*
- REAL(DP), allocatable [COUPLING\\_COEFFICIENTS](#)  
*COUPLING\_COEFFICIENTS(i). The i'th coupling coefficient for solver column mapping.*
- INTEGER(INTG) [JACOBIAN\\_COL\\_NUMBER](#)  
*The Jacobian column number in the equations set that the solver column is mapped to.*
- REAL(DP) [JACOBIAN\\_COUPLING\\_COEFFICIENT](#)  
*The coupling coefficient for the solver column to Jacobian column mapping.*

### 7.304.1 Detailed Description

Contains information about the mapping from a solver matrix column to equations matrices and variables.  
 Definition at line 1553 of file types.f90.

### 7.304.2 Member Data Documentation

#### 7.304.2.1 REAL(DP),allocatable TYPES::SOLVER\_COL\_TO\_EQUATIONS\_MAP\_- TYPE::COUPLING\_COEFFICIENTS

COUPLING\_COEFFICIENTS(i). The i'th coupling coefficient for solver column mapping.  
 Definition at line 1557 of file types.f90.

#### 7.304.2.2 INTEGER(INTG),allocatable TYPES::SOLVER\_COL\_TO\_EQUATIONS\_MAP\_- TYPE::EQUATIONS\_COL\_NUMBERS

EQUATIONS\_COL\_NUMBERS(i). The i'th equations column number in the equation set the solver column is mapped to.  
 Definition at line 1556 of file types.f90.

**7.304.2.3 INTEGER(INTG),allocatable TYPES::SOLVER\_COL\_TO\_EQUATIONS\_MAP\_-  
TYPE::EQUATIONS\_MATRIX\_NUMBERS**

EQUATIONS\_MATRIX\_NUMBERS(i). The i'th equations matrix number in the equations that the solver column is mapped to.

Definition at line 1555 of file types.f90.

**7.304.2.4 INTEGER(INTG) TYPES::SOLVER\_COL\_TO\_EQUATIONS\_MAP\_-  
TYPE::JACOBIAN\_COL\_NUMBER**

The Jacobian column number in the equations set that the solver column is mapped to.

Definition at line 1559 of file types.f90.

**7.304.2.5 REAL(DP) TYPES::SOLVER\_COL\_TO\_EQUATIONS\_MAP\_TYPE::JACOBIAN\_-  
COUPLING\_COEFFICIENT**

The coupling coefficient for the solver column to Jacobian column mapping.

Definition at line 1560 of file types.f90.

**7.304.2.6 INTEGER(INTG) TYPES::SOLVER\_COL\_TO\_EQUATIONS\_MAP\_-  
TYPE::NUMBER\_OF\_LINEAR\_EQUATIONS\_MATRICES**

The number of linear equations matrices the solver column is mapped to in this equations set.

Definition at line 1554 of file types.f90.

## 7.305 TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SET\_MAP\_- TYPE Struct Reference

Contains information about the mappings from a solver matrix to the equations in an equations set.

Collaboration diagram for TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SET\_MAP\_TYPE:

### Public Attributes

- TYPE(EQUATIONS\_TYPE), pointer EQUATIONS
- TYPE(SOLVER\_COL\_TO\_EQUATIONS\_MAP\_TYPE), allocatable SOLVER\_COL\_TO\_EQUATIONS\_MAPS

*SOLVER\_COL\_TO\_EQUATIONS\_MAPS(col\_idx). The mappings from the col\_idx'th column of the solver matrix to the equations in the equations set.*

### 7.305.1 Detailed Description

Contains information about the mappings from a solver matrix to the equations in an equations set.

Definition at line 1574 of file types.f90.

### 7.305.2 Member Data Documentation

#### 7.305.2.1 TYPE(EQUATIONS\_TYPE),pointer TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SET\_MAP\_TYPE::EQUATIONS

Definition at line 1575 of file types.f90.

#### 7.305.2.2 TYPE(SOLVER\_COL\_TO\_EQUATIONS\_MAP\_TYPE),allocatable TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SET\_MAP\_TYPE::SOLVER\_COL\_TO\_EQUATIONS\_MAPS

*SOLVER\_COL\_TO\_EQUATIONS\_MAPS(col\_idx). The mappings from the col\_idx'th column of the solver matrix to the equations in the equations set.*

Definition at line 1576 of file types.f90.

## 7.306 TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SETS\_MAP\_TYPE Struct Reference

Contains information on the mappings from a solver matrix to equations sets.

Collaboration diagram for TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SETS\_MAP\_TYPE:

### Public Attributes

- INTEGER(INTG) **SOLVER\_MATRIX\_NUMBER**  
*The number of this solver matrix.*
- TYPE(**SOLUTION\_MAPPING\_TYPE**), pointer **SOLUTION\_MAPPING**  
*A pointer to the solution mapping for this solver matrix mapping.*
- TYPE(**SOLVER\_MATRIX\_TYPE**), pointer **SOLVER\_MATRIX**  
*A pointer to the solver matrix being mapped.*
- INTEGER(INTG) **NUMBER\_OF\_COLUMNS**  
*The number of columns in this distributed solver matrix.*
- TYPE(**SOLVER\_COL\_TO\_EQUATIONS\_SET\_MAP\_TYPE**), allocatable **SOLVER\_COL\_TO\_EQUIVATIONS\_SET\_MAPS**  
*SOLVER\_TO\_EQUIVATIONS\_SET\_MAP(equations\_set\_idx). The solver columns to equations matrix maps for the col\_idx'th column set.*
- INTEGER(INTG) **NUMBER\_OF\_DOFS**  
*The number of local dofs (excluding ghost values) in the solver vector associated with this solver matrix.*
- INTEGER(INTG) **TOTAL\_NUMBER\_OF\_DOFS**  
*The number of local dofs (including ghost values) in the solver vector associated with this solver matrix.*
- INTEGER(INTG) **NUMBER\_OF\_GLOBAL\_DOFS**  
*The number of global dofs in the solver vector associated with this solver matrix.*
- TYPE(**SOLVER\_DOF\_TO\_VARIABLE\_MAP\_TYPE**), allocatable **SOLVER\_DOF\_TO\_VARIABLE\_MAPS**  
*SOLVER\_DOF\_TO\_EQUIVATIONS\_MAPS(dof\_idx). The mappings from the dof\_idx'th solver dof to the field variables in the equations set.*
- TYPE(**DOMAIN\_MAPPING\_TYPE**), pointer **COLUMN\_DOFS\_MAPPING**  
*The domain mapping for solver matrix column dofs.*

### 7.306.1 Detailed Description

Contains information on the mappings from a solver matrix to equations sets.

Definition at line 1580 of file types.f90.

## 7.306.2 Member Data Documentation

### 7.306.2.1 **TYPE(DOMAIN\_MAPPING\_TYPE),pointer TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SETS\_MAP\_TYPE::COLUMN\_DOFS\_MAPPING**

The domain mapping for solver matrix column dofs.

Definition at line 1591 of file types.f90.

### 7.306.2.2 **INTEGER(INTG) TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SETS\_MAP\_TYPE::NUMBER\_OF\_COLUMNS**

The number of columns in this distributed solver matrix.

Definition at line 1584 of file types.f90.

### 7.306.2.3 **INTEGER(INTG) TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SETS\_MAP\_TYPE::NUMBER\_OF\_DOFS**

The number of local dofs (excluding ghost values) in the solver vector associated with this solver matrix.

Definition at line 1586 of file types.f90.

### 7.306.2.4 **INTEGER(INTG) TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SETS\_MAP\_TYPE::NUMBER\_OF\_GLOBAL\_DOFS**

The number of global dofs in the solver vector associated with this solver matrix.

Definition at line 1588 of file types.f90.

### 7.306.2.5 **TYPE(SOLUTION\_MAPPING\_TYPE),pointer TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SETS\_MAP\_TYPE::SOLUTION\_MAPPING**

A pointer to the solution mapping for this solver matrix mapping.

Definition at line 1582 of file types.f90.

### 7.306.2.6 **TYPE(SOLVER\_COL\_TO\_EQUATIONS\_SET\_MAP\_TYPE),allocatable TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SETS\_MAP\_TYPE::SOLVER\_COL\_TO\_EQUATIONS\_SET\_MAPS**

SOLVER\_TO\_EQUATIONS\_SET\_MAP(equations\_set\_idx). The solver columns to equations matrix maps for the col\_idx'th column set.

Definition at line 1585 of file types.f90.

### 7.306.2.7 **TYPE(SOLVER\_DOF\_TO\_VARIABLE\_MAP\_TYPE),allocatable TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SETS\_MAP\_TYPE::SOLVER\_DOF\_TO\_VARIABLE\_MAPS**

SOLVER\_DOF\_TO\_EQUATIONS\_MAPS(dof\_idx). The mappings from the dof\_idx'th solver dof to the field variables in the equations set.

Definition at line 1590 of file types.f90.

**7.306.2.8 TYPE(SOLVER\_MATRIX\_TYPE),pointer TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SETS\_MAP\_TYPE::SOLVER\_MATRIX**

A pointer to the solver matrix being mappind.

Definition at line 1583 of file types.f90.

**7.306.2.9 INTEGER(INTG) TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SETS\_MAP\_TYPE::SOLVER\_MATRIX\_NUMBER**

The number of this solver matrix.

Definition at line 1581 of file types.f90.

**7.306.2.10 INTEGER(INTG) TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SETS\_MAP\_TYPE::TOTAL\_NUMBER\_OF\_DOFS**

The number of local dofs (including ghost values) in the solver vector associated with this solver matrix.

Definition at line 1587 of file types.f90.

## 7.307 TYPES::SOLVER\_DOF\_TO\_VARIABLE\_MAP\_TYPE Struct Reference

Contains information about mapping the solver dof to the field variable dofs in the equations set.

Collaboration diagram for TYPES::SOLVER\_DOF\_TO\_VARIABLE\_MAP\_TYPE:

### Public Attributes

- INTEGER(INTG) [NUMBER\\_OF\\_EQUATIONS\\_SETS](#)  
*The number of equations sets this column is mapped to.*
- INTEGER(INTG), allocatable [EQUATIONS\\_SET\\_INDICES](#)  
*EQUATIONS\_SET\_INDICES(i). The equations set index of the i'th equations set that this solver column is mapped to.*
- TYPE([FIELD\\_VARIABLE\\_PTR\\_TYPE](#)), allocatable [VARIABLE](#)  
*VARIABLE(i). VARIABLE(i)PTR is a pointer to the field variable that the column is mapped to in the i'th equation set.*
- INTEGER(INTG), allocatable [VARIABLE\\_DOF](#)  
*VARIABLE\_DOF(i). The variable dof number that the column is mapped to in the i'th equations set.*
- REAL(DP), allocatable [VARIABLE\\_COEFFICIENT](#)  
*VARIABLE\_COEFFICIENT(i). The multiplicative coefficient for the mapping to the i'th equations set.*
- REAL(DP), allocatable [ADDITIVE\\_CONSTANT](#)  
*ADDITIVE\_CONSTANT(i). The additive constant for the mapping to the i'th equations set.*

### 7.307.1 Detailed Description

Contains information about mapping the solver dof to the field variable dofs in the equations set.

Definition at line 1564 of file types.f90.

### 7.307.2 Member Data Documentation

#### 7.307.2.1 REAL(DP),allocatable TYPES::SOLVER\_DOF\_TO\_VARIABLE\_MAP\_TYPE::ADDITIVE\_CONSTANT

ADDITIVE\_CONSTANT(i). The additive constant for the mapping to the i'th equations set.

Definition at line 1570 of file types.f90.

#### 7.307.2.2 INTEGER(INTG),allocatable TYPES::SOLVER\_DOF\_TO\_VARIABLE\_MAP\_TYPE::EQUATIONS\_SET\_INDICES

EQUATIONS\_SET\_INDICES(i). The equations set index of the i'th equations set that this solver column is mapped to.

Definition at line 1566 of file types.f90.

**7.307.2.3 INTEGER(INTG) TYPES::SOLVER\_DOF\_TO\_VARIABLE\_MAP\_-  
TYPE::NUMBER\_OF\_EQUATIONS\_SETS**

The number of equations sets this column is mapped to.

Definition at line 1565 of file types.f90.

**7.307.2.4 TYPE(FIELD\_VARIABLE\_PTR\_TYPE),allocatable TYPES::SOLVER\_DOF\_TO\_-  
VARIABLE\_MAP\_TYPE::VARIABLE**

VARIABLE(i).VARIABLE(i)PTR is a pointer to the field variable that the column is mapped to in the i'th equation set.

Definition at line 1567 of file types.f90.

**7.307.2.5 REAL(DP),allocatable TYPES::SOLVER\_DOF\_TO\_VARIABLE\_MAP\_-  
TYPE::VARIABLE\_COEFFICIENT**

VARIABLE\_COEFFICIENT(i). The mulitplicative coefficient for the mapping to the i'th equations set.

Definition at line 1569 of file types.f90.

**7.307.2.6 INTEGER(INTG),allocatable TYPES::SOLVER\_DOF\_TO\_VARIABLE\_MAP\_-  
TYPE::VARIABLE\_DOF**

VARIABLE\_DOF(i). The variable dof number that the column is mapped to in the i'th equations set.

Definition at line 1568 of file types.f90.

## 7.308 TYPES::SOLVER\_JACOBIAN\_TYPE Struct Reference

Contains information on the solver Jacobian for nonlinear problems.

Collaboration diagram for TYPES::SOLVER\_JACOBIAN\_TYPE:

### Public Attributes

- TYPE(SOLVER\_MATRICES\_TYPE), pointer SOLVER\_MATRICES  
*A pointer to the solver matrices for this solver Jacobian.*
- LOGICAL UPDATE\_JACOBIAN  
*Is .TRUE. if the solver Jacobian is to be updated.*
- INTEGER(INTG) STORAGE\_TYPE  
*The storage type for the solver Jacobian.*
- INTEGER(INTG) NUMBER\_OF\_COLUMNS  
*The number of columns in the distributed solver Jacobian.*
- TYPE(DISTRIBUTED\_MATRIX\_TYPE), pointer JACOBIAN  
*A pointer to the distributed solver Jacobian.*

### 7.308.1 Detailed Description

Contains information on the solver Jacobian for nonlinear problems.

Definition at line 1340 of file types.f90.

### 7.308.2 Member Data Documentation

#### 7.308.2.1 TYPE(DISTRIBUTED\_MATRIX\_TYPE),pointer TYPES::SOLVER\_JACOBIAN\_- TYPE::JACOBIAN

A pointer to the distributed solver Jacobian.

Definition at line 1345 of file types.f90.

#### 7.308.2.2 INTEGER(INTG) TYPES::SOLVER\_JACOBIAN\_TYPE::NUMBER\_OF\_COLUMNS

The number of columns in the distributed solver Jacobian.

Definition at line 1344 of file types.f90.

#### 7.308.2.3 TYPE(SOLVER\_MATRICES\_TYPE),pointer TYPES::SOLVER\_JACOBIAN\_- TYPE::SOLVER\_MATRICES

A pointer to the solver matrices for this solver Jacobian.

Definition at line 1341 of file types.f90.

**7.308.2.4 INTEGER(INTG) TYPES::SOLVER\_JACOBIAN\_TYPE::STORAGE\_TYPE**

The storage type for the solver Jacobian.

Definition at line 1343 of file types.f90.

**7.308.2.5 LOGICAL TYPES::SOLVER\_JACOBIAN\_TYPE::UPDATE\_JACOBIAN**

Is .TRUE. if the solver Jacobian is to be updated.

Definition at line 1342 of file types.f90.

## 7.309 TYPES::SOLVER\_MATRICES\_TYPE Struct Reference

Contains information on the solver matrices and rhs vector.

Collaboration diagram for TYPES::SOLVER\_MATRICES\_TYPE:

### Public Attributes

- TYPE(SOLVER\_TYPE), pointer SOLVER  
*A pointer to the problem solver.*
- LOGICAL SOLVER\_MATRICES\_FINISHED  
*Is .TRUE. if the solver matrices have finished being created, .FALSE. if not.*
- TYPE(SOLUTION\_MAPPING\_TYPE), pointer SOLUTION\_MAPPING  
*A pointer to the solution mapping for these solver matrices.*
- INTEGER(INTG) NUMBER\_OF\_ROWS  
*The number of (local) rows in the distributed solution matrix for this computational node.*
- INTEGER(INTG) NUMBER\_OF\_GLOBAL\_ROWS  
*The number of global rows in the distributed solution matrix.*
- INTEGER(INTG) LIBRARY\_TYPE  
*The library type for the solver matrices.*
- INTEGER(INTG) NUMBER\_OF\_MATRICES  
*The number of solver matrices defined for the problem.*
- TYPE(SOLVER\_MATRIX\_PTR\_TYPE), allocatable MATRICES  
*MATRICES(matrix\_idx)PTR contains the information on the matrix\_idx'th solver matrix.*
- LOGICAL UPDATE\_RESIDUAL  
*Is .TRUE. if the residual vector is to be updated.*
- TYPE(DISTRIBUTED\_VECTOR\_TYPE), pointer RESIDUAL  
*A pointer to the distributed residual vector for nonlinear problems.*
- LOGICAL UPDATE\_RHS\_VECTOR  
*Is .TRUE. if the RHS vector is to be updated.*
- TYPE(DISTRIBUTED\_VECTOR\_TYPE), pointer RHS\_VECTOR  
*A pointer to the distributed RHS vector for the solver matrices.*

### 7.309.1 Detailed Description

Contains information on the solver matrices and rhs vector.

Definition at line 1349 of file types.f90.

## 7.309.2 Member Data Documentation

### 7.309.2.1 INTEGER(INTG) TYPES::SOLVER\_MATRICES\_TYPE::LIBRARY\_TYPE

The library type for the solver matrices.

Definition at line 1355 of file types.f90.

### 7.309.2.2 TYPE(SOLVER\_MATRIX\_PTR\_TYPE),allocatable TYPES::SOLVER\_MATRICES\_TYPE::MATRICES

MATRICES(matrix\_idx)PTR contains the information on the matrix\_idx'th solver matrix.

Definition at line 1358 of file types.f90.

### 7.309.2.3 INTEGER(INTG) TYPES::SOLVER\_MATRICES\_TYPE::NUMBER\_OF\_GLOBAL\_ROWS

The number of global rows in the distributed solution matrix.

Definition at line 1354 of file types.f90.

### 7.309.2.4 INTEGER(INTG) TYPES::SOLVER\_MATRICES\_TYPE::NUMBER\_OF\_MATRICES

The number of solver matrices defined for the problem.

Definition at line 1357 of file types.f90.

### 7.309.2.5 INTEGER(INTG) TYPES::SOLVER\_MATRICES\_TYPE::NUMBER\_OF\_ROWS

The number of (local) rows in the distributed solution matrix for this computational node.

Definition at line 1353 of file types.f90.

### 7.309.2.6 TYPE(DISTRIBUTED\_VECTOR\_TYPE),pointer TYPES::SOLVER\_MATRICES\_TYPE::RESIDUAL

A pointer to the distributed residual vector for nonlinear problems.

Definition at line 1361 of file types.f90.

### 7.309.2.7 TYPE(DISTRIBUTED\_VECTOR\_TYPE),pointer TYPES::SOLVER\_MATRICES\_TYPE::RHS\_VECTOR

A pointer to the distributed RHS vector for the solver matrices.

Definition at line 1364 of file types.f90.

### 7.309.2.8 TYPE(SOLUTION\_MAPPING\_TYPE),pointer TYPES::SOLVER\_MATRICES\_TYPE::SOLUTION\_MAPPING

A pointer to the solution mapping for these solver matrices.

Definition at line 1352 of file types.f90.

#### **7.309.2.9 TYPE(SOLVER\_TYPE),pointer TYPES::SOLVER\_MATRICES\_TYPE::SOLVER**

A pointer to the problem solver.

Definition at line 1350 of file types.f90.

#### **7.309.2.10 LOGICAL TYPES::SOLVER\_MATRICES\_TYPE::SOLVER\_MATRICES\_-FINISHED**

Is .TRUE. if the solver matrices have finished being created, .FALSE. if not.

Definition at line 1351 of file types.f90.

#### **7.309.2.11 LOGICAL TYPES::SOLVER\_MATRICES\_TYPE::UPDATE\_RESIDUAL**

Is .TRUE. if the residual vector is to be updated.

Definition at line 1360 of file types.f90.

#### **7.309.2.12 LOGICAL TYPES::SOLVER\_MATRICES\_TYPE::UPDATE\_RHS\_VECTOR**

Is .TRUE. if the RHS vector is to be updated.

Definition at line 1363 of file types.f90.

## 7.310 TYPES::SOLVER\_MATRIX\_PTR\_TYPE Struct Reference

Collaboration diagram for TYPES::SOLVER\_MATRIX\_PTR\_TYPE:

### Public Attributes

- TYPE(SOLVER\_MATRIX\_TYPE), pointer PTR

#### 7.310.1 Detailed Description

Definition at line 1335 of file types.f90.

#### 7.310.2 Member Data Documentation

##### 7.310.2.1 TYPE(SOLVER\_MATRIX\_TYPE),pointer TYPES::SOLVER\_MATRIX\_PTR\_- TYPE::PTR

Definition at line 1336 of file types.f90.

## 7.311 TYPES::SOLVER\_MATRIX\_TYPE Struct Reference

Contains information on the solver matrix.

Collaboration diagram for TYPES::SOLVER\_MATRIX\_TYPE:

### Public Attributes

- INTEGER(INTG) **MATRIX\_NUMBER**  
*The number of the solver matrix.*
- TYPE(SOLVER\_MATRICES\_TYPE), pointer **SOLVER\_MATRICES**  
*A pointer to the solver matrices for this solver matrix.*
- LOGICAL **UPDATE\_MATRIX**  
*Is .TRUE. if the solver matrix is to be updated.*
- INTEGER(INTG) **STORAGE\_TYPE**  
*The storage type for the solver matrix.*
- INTEGER(INTG) **NUMBER\_OF\_COLUMNS**  
*The number of columns in the distributed solver matrix.*
- TYPE(DISTRIBUTED\_VECTOR\_TYPE), pointer **SOLVER\_VECTOR**  
*A pointer to the distributed solver vector associated with the matrix.*
- TYPE(DISTRIBUTED\_MATRIX\_TYPE), pointer **MATRIX**  
*A pointer to the distributed solver matrix data.*

### 7.311.1 Detailed Description

Contains information on the solver matrix.

Definition at line 1325 of file types.f90.

### 7.311.2 Member Data Documentation

#### 7.311.2.1 TYPE(DISTRIBUTED\_MATRIX\_TYPE),pointer TYPES::SOLVER\_MATRIX\_- TYPE::MATRIX

A pointer to the distributed solver matrix data.

Definition at line 1332 of file types.f90.

#### 7.311.2.2 INTEGER(INTG) TYPES::SOLVER\_MATRIX\_TYPE::MATRIX\_NUMBER

The number of the solver matrix.

Definition at line 1326 of file types.f90.

**7.311.2.3 INTEGER(INTG) TYPES::SOLVER\_MATRIX\_TYPE::NUMBER\_OF\_COLUMNS**

The number of columns in the distributed solver matrix.

Definition at line 1330 of file types.f90.

**7.311.2.4 TYPE(SOLVER\_MATRICES\_TYPE),pointer TYPES::SOLVER\_MATRIX\_-  
TYPE::SOLVER\_MATRICES**

A pointer to the solver matrices for this solver matrix.

Definition at line 1327 of file types.f90.

**7.311.2.5 TYPE(DISTRIBUTED\_VECTOR\_TYPE),pointer TYPES::SOLVER\_MATRIX\_-  
TYPE::SOLVER\_VECTOR**

A pointer to the distributed solver vector associated with the matrix.

Definition at line 1331 of file types.f90.

**7.311.2.6 INTEGER(INTG) TYPES::SOLVER\_MATRIX\_TYPE::STORAGE\_TYPE**

The storage type for the solver matrix.

Definition at line 1329 of file types.f90.

**7.311.2.7 LOGICAL TYPES::SOLVER\_MATRIX\_TYPE::UPDATE\_MATRIX**

Is .TRUE. if the solver matrix is to be updated.

Definition at line 1328 of file types.f90.

## 7.312 TYPES::SOLVER\_ROW\_TO\_EQUATIONS\_SET\_MAP\_- TYPE Struct Reference

Contains information on the mappings from a solver row to the equations rows.

### Public Attributes

- INTEGER(INTG) **NUMBER\_OF\_ROWS**  
*The number of rows the solver row is mapped to.*
- INTEGER(INTG), allocatable **EQUATIONS\_SET**  
*EQUATIONS\_SET(i). The equation set index of i'th row that the solver row is mapped to.*
- INTEGER(INTG), allocatable **EQUATIONS\_ROW\_NUMBER**  
*EQUATIONS\_ROW\_NUMBER(i). The i'th equations row number in the equation set the solver row is mapped to.*
- REAL(DP), allocatable **COUPLING\_COEFFICIENTS**  
*COUPLING\_COEFFICIENTS(i). The i'th coupling coefficient for solver row mapping.*

### 7.312.1 Detailed Description

Contains information on the mappings from a solver row to the equations rows.

Definition at line 1595 of file types.f90.

### 7.312.2 Member Data Documentation

#### 7.312.2.1 REAL(DP),allocatable TYPES::SOLVER\_ROW\_TO\_EQUATIONS\_SET\_MAP\_- TYPE::COUPLING\_COEFFICIENTS

COUPLING\_COEFFICIENTS(i). The i'th coupling coefficient for solver row mapping.

Definition at line 1599 of file types.f90.

#### 7.312.2.2 INTEGER(INTG),allocatable TYPES::SOLVER\_ROW\_TO\_EQUATIONS\_SET\_- MAP\_TYPE::EQUATIONS\_ROW\_NUMBER

EQUATIONS\_ROW\_NUMBER(i). The i'th equations row number in the equation set the solver row is mapped to.

Definition at line 1598 of file types.f90.

#### 7.312.2.3 INTEGER(INTG),allocatable TYPES::SOLVER\_ROW\_TO\_EQUATIONS\_SET\_- MAP\_TYPE::EQUATIONS\_SET

EQUATIONS\_SET(i). The equation set index of i'th row that the solver row is mapped to.

Definition at line 1597 of file types.f90.

**7.312.2.4 INTEGER(INTG) TYPES::SOLVER\_ROW\_TO\_EQUATIONS\_SET\_MAP\_-  
TYPE::NUMBER\_OF\_ROWS**

The number of rows the solver row is mapped to.

Definition at line 1596 of file types.f90.

## 7.313 TYPES::SOLVER\_TYPE Struct Reference

Contains information on the type of solver to be used.

Collaboration diagram for TYPES::SOLVER\_TYPE:

### Public Attributes

- TYPE(SOLUTION\_TYPE), pointer SOLUTION  
*A pointer to the problem solution.*
- LOGICAL SOLVER\_FINISHED  
*Is .TRUE. if the problem solver has finished being created, .FALSE. if not.*
- TYPE(SOLUTION\_MAPPING\_TYPE), pointer SOLUTION\_MAPPING  
*A pointer to the problem solution.*
- INTEGER(INTG) SOLVE\_TYPE  
*The type of the problem solver.*
- INTEGER(INTG) OUTPUT\_TYPE  
*The type of output required.*
- INTEGER(INTG) SPARSITY\_TYPE  
*The type of sparsity to use in the solver matrices.*
- TYPE(LINEAR\_SOLVER\_TYPE), pointer LINEAR\_SOLVER  
*A pointer to the linear solver information.*
- TYPE(NONLINEAR\_SOLVER\_TYPE), pointer NONLINEAR\_SOLVER  
*A pointer to the nonlinear solver information.*
- TYPE(TIME\_INTEGRATION\_SOLVER\_TYPE), pointer TIME\_INTEGRATION\_SOLVER  
*A pointer to the time integration solver information.*
- TYPE(EIGENPROBLEM\_SOLVER\_TYPE), pointer EIGENPROBLEM\_SOLVER  
*A pointer to the eigenproblem solver information.*
- TYPE(SOLVER\_MATRICES\_TYPE), pointer SOLVER\_MATRICES  
*A pointer to the solver matrices for the problem.*

### 7.313.1 Detailed Description

Contains information on the type of solver to be used.

Definition at line 1451 of file types.f90.

### 7.313.2 Member Data Documentation

#### 7.313.2.1 TYPE(EIGENPROBLEM\_SOLVER\_TYPE),pointer TYPES::SOLVER\_- TYPE::EIGENPROBLEM\_SOLVER

A pointer to the eigenproblem solver information.

Definition at line 1461 of file types.f90.

#### 7.313.2.2 TYPE(LINEAR\_SOLVER\_TYPE),pointer TYPES::SOLVER\_TYPE::LINEAR\_- SOLVER

A pointer to the linear solver information.

Definition at line 1458 of file types.f90.

#### 7.313.2.3 TYPE(NONLINEAR\_SOLVER\_TYPE),pointer TYPES::SOLVER\_- TYPE::NONLINEAR\_SOLVER

A pointer to the nonlinear solver information.

Definition at line 1459 of file types.f90.

#### 7.313.2.4 INTEGER(INTG) TYPES::SOLVER\_TYPE::OUTPUT\_TYPE

The type of output required.

**See also:**

[SOLVER\\_ROUTINES::OutputTypes](#),[SOLVER\\_ROUTINES](#)

Definition at line 1456 of file types.f90.

#### 7.313.2.5 TYPE(SOLUTION\_TYPE),pointer TYPES::SOLVER\_TYPE::SOLUTION

A pointer to the problem solution.

Definition at line 1452 of file types.f90.

#### 7.313.2.6 TYPE(SOLUTION\_MAPPING\_TYPE),pointer TYPES::SOLVER\_- TYPE::SOLUTION\_MAPPING

A pointer to the problem solution.

Definition at line 1454 of file types.f90.

#### 7.313.2.7 INTEGER(INTG) TYPES::SOLVER\_TYPE::SOLVE\_TYPE

The type of the problem solver.

**See also:**

[SOLVER\\_ROUTINES::SolverTypes](#),[SOLVER\\_ROUTINES](#)

Definition at line 1455 of file types.f90.

#### **7.313.2.8 LOGICAL TYPES::SOLVER\_TYPE::SOLVER\_FINISHED**

Is .TRUE. if the problem solver has finished being created, .FALSE. if not.

Definition at line 1453 of file types.f90.

#### **7.313.2.9 TYPE(SOLVER\_MATRICES\_TYPE),pointer TYPES::SOLVER\_TYPE::SOLVER\_MATRICES**

A pointer to the solver matrices for the problem.

Definition at line 1462 of file types.f90.

#### **7.313.2.10 INTEGER(INTG) TYPES::SOLVER\_TYPE::SPARSITY\_TYPE**

The type of sparsity to use in the solver matrices.

**See also:**

SOLVER\_ROUTINES\_SparsityTypes,[SOLVER\\_ROUTINES](#)

Definition at line 1457 of file types.f90.

#### **7.313.2.11 TYPE(TIME\_INTEGRATION\_SOLVER\_TYPE),pointer TYPES::SOLVER\_TYPE::TIME\_INTEGRATION\_SOLVER**

A pointer to the time integration solver information.

Definition at line 1460 of file types.f90.

## 7.314 TYPES::TIME\_INTEGRATION\_SOLVER\_TYPE Struct Reference

Contains information for a time integration solver.

Collaboration diagram for TYPES::TIME\_INTEGRATION\_SOLVER\_TYPE:

### Public Attributes

- TYPE([SOLVER\\_TYPE](#)), pointer **SOLVER**  
*A pointer to the problem\_solver.*
- INTEGER(INTG) **SOLVER\_LIBRARY**  
*The library type for the time integration solver.*

#### 7.314.1 Detailed Description

Contains information for a time integration solver.

Definition at line 1439 of file types.f90.

#### 7.314.2 Member Data Documentation

##### 7.314.2.1 TYPE(SOLVER\_TYPE),pointer TYPES::TIME\_INTEGRATION\_SOLVER\_- TYPE::SOLVER

A pointer to the problem\_solver.

Definition at line 1440 of file types.f90.

##### 7.314.2.2 INTEGER(INTG) TYPES::TIME\_INTEGRATION\_SOLVER\_TYPE::SOLVER\_- LIBRARY

The library type for the time integration solver.

#### See also:

[SOLVER\\_ROUTINES::SolverLibraries](#),[SOLVER\\_ROUTINES](#)

Definition at line 1441 of file types.f90.

## 7.315 TYPES::VARIABLE\_TO\_EQUATIONS\_COLUMN\_MAP\_- TYPE Struct Reference

Contains the information about the mapping of a variable DOF to an equations matrix column.

### Public Attributes

- INTEGER(INTG), allocatable [COLUMN\\_DOF](#)

*COLUMN\_DOF(dof\_idx).* The equations column number for this equations matrix that the dof\_idx'th variable DOF is mapped to.

### 7.315.1 Detailed Description

Contains the information about the mapping of a variable DOF to an equations matrix column.

Definition at line 1051 of file types.f90.

### 7.315.2 Member Data Documentation

#### 7.315.2.1 INTEGER(INTG),allocatable TYPES::VARIABLE\_TO\_EQUATIONS\_COLUMN\_- MAP\_TYPE::COLUMN\_DOF

COLUMN\_DOF(dof\_idx). The equations column number for this equations matrix that the dof\_idx'th variable DOF is mapped to.

Definition at line 1052 of file types.f90.

## 7.316 TYPES::VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_- MAP\_TYPE Struct Reference

Contains the mapping for a dependent variable type to the nonlinear Jacobian matrix.

Collaboration diagram for TYPES::VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_MAP\_TYPE:

### Public Attributes

- INTEGER(INTG) [VARIABLE\\_TYPE](#)  
*The variable type for this variable to equations matrices map.*
- TYPE([FIELD\\_VARIABLE\\_TYPE](#)), pointer [VARIABLE](#)  
*A pointer to the field variable for this variable to equations matrices map.*
- INTEGER(INTG), allocatable [DOF\\_TO\\_COLUMNS\\_MAP](#)  
*DOF\_TO\_COLUMNS\_MAP(dof\_idx). The Jacobian column number for dof\_idx'th variable dof.*
- INTEGER(INTG), allocatable [DOF\\_TO\\_ROWS\\_MAP](#)  
*DOF\_TO\_ROWS\_MAP(dof\_idx). The row number that the dof\_idx'th variable dof is mapped to.*

### 7.316.1 Detailed Description

Contains the mapping for a dependent variable type to the nonlinear Jacobian matrix.

Definition at line 1099 of file types.f90.

### 7.316.2 Member Data Documentation

#### 7.316.2.1 INTEGER(INTG),allocatable TYPES::VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_- MAP\_TYPE::DOF\_TO\_COLUMNS\_MAP

DOF\_TO\_COLUMNS\_MAP(dof\_idx). The Jacobian column number for dof\_idx'th variable dof.

Definition at line 1102 of file types.f90.

#### 7.316.2.2 INTEGER(INTG),allocatable TYPES::VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_- MAP\_TYPE::DOF\_TO\_ROWS\_MAP

DOF\_TO\_ROWS\_MAP(dof\_idx). The row number that the dof\_idx'th variable dof is mapped to.

Definition at line 1103 of file types.f90.

#### 7.316.2.3 TYPE(FIELD\_VARIABLE\_TYPE),pointer TYPES::VARIABLE\_TO\_EQUATIONS\_- JACOBIAN\_MAP\_TYPE::VARIABLE

A pointer to the field variable for this variable to equations matrices map.

Definition at line 1101 of file types.f90.

**7.316.2.4 INTEGER(INTG) TYPES::VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_MAP\_-  
TYPE::VARIABLE\_TYPE**

The variable type for this variable to equations matrices map.

Definition at line 1100 of file types.f90.

## 7.317 TYPES::VARIABLE\_TO\_EQUATIONS\_MATRICES\_- MAP\_TYPE Struct Reference

Contains the mapping for a dependent variable type to the equations matrices.

Collaboration diagram for TYPES::VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP\_TYPE:

### Public Attributes

- INTEGER(INTG) [VARIABLE\\_INDEX](#)  
*The variable index for this variable to equations matrices map.*
- INTEGER(INTG) [VARIABLE\\_TYPE](#)  
*The variable type for this variable to equations matrices map.*
- TYPE([FIELD\\_VARIABLE\\_TYPE](#)), pointer [VARIABLE](#)  
*A pointer to the field variable for this variable to equations matrices map.*
- INTEGER(INTG) [NUMBER\\_OF\\_LINEAR\\_EQUATIONS\\_MATRICES](#)  
*The number of linear equations matrices this variable type is mapped to. If the number is -1 the variable is mapped to the RHS vector. If the number is zero then this variable type is not involved in the equations set and the rest of the type is not allocated.*
- INTEGER(INTG), allocatable [EQUATIONS\\_MATRIX\\_NUMBERS](#)  
*EQUATIONS\_MATRIX\_NUMBERS(i). The equations matrix number for the i'th matrix that this variable type is mapped to.*
- TYPE([VARIABLE\\_TO\\_EQUATIONS\\_COLUMN\\_MAP\\_TYPE](#)), allocatable [DOF\\_TO\\_COLUMNS\\_MAPS](#)  
*DOF\_TO\_COLUMNS\_MAPS(i). The variable dof to equations columns for the i'th equations matrix.*
- INTEGER(INTG), allocatable [DOF\\_TO\\_ROWS\\_MAP](#)  
*DOF\_TO\_ROWS\_MAP(dof\_idx). The row number that the dof\_idx'th variable dof is mapped to.*

### 7.317.1 Detailed Description

Contains the mapping for a dependent variable type to the equations matrices.

Definition at line 1056 of file types.f90.

### 7.317.2 Member Data Documentation

#### 7.317.2.1 TYPE(VARIABLE\_TO\_EQUATIONS\_COLUMN\_MAP\_TYPE),allocatable TYPES::VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP\_TYPE::DOF\_TO\_- COLUMNS\_MAPS

DOF\_TO\_COLUMNS\_MAPS(i). The variable dof to equations columns for the i'th equations matrix.

Definition at line 1062 of file types.f90.

**7.317.2.2 INTEGER(INTG),allocatable TYPES::VARIABLE\_TO\_EQUATIONS\_MATRICES\_-  
MAP\_TYPE::DOF\_TO\_ROWS\_MAP**

DOF\_TO\_ROWS\_MAP(dof\_idx). The row number that the dof\_idx'th variable dof is mapped to.

Definition at line 1063 of file types.f90.

**7.317.2.3 INTEGER(INTG),allocatable TYPES::VARIABLE\_TO\_EQUATIONS\_MATRICES\_-  
MAP\_TYPE::EQUATIONS\_MATRIX\_NUMBERS**

EQUATIONS\_MATRIX\_NUMBERS(i). The equations matrix number for the i'th matrix that this variable type is mapped to.

Definition at line 1061 of file types.f90.

**7.317.2.4 INTEGER(INTG) TYPES::VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP\_-  
TYPE::NUMBER\_OF\_LINEAR\_EQUATIONS\_MATRICES**

The number of linear equations matrices this variable type is mapped to. If the number is -1 the variable is mapped to the RHS vector. If the number is zero then this variable type is not involved in the equations set and the rest of the type is not allocated.

Definition at line 1060 of file types.f90.

**7.317.2.5 TYPE(FIELD\_VARIABLE\_TYPE),pointer TYPES::VARIABLE\_TO\_EQUATIONS\_-  
MATRICES\_MAP\_TYPE::VARIABLE**

A pointer to the field variable for this variable to equations matrices map.

Definition at line 1059 of file types.f90.

**7.317.2.6 INTEGER(INTG) TYPES::VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP\_-  
TYPE::VARIABLE\_INDEX**

The variable index for this variable to equations matrices map.

Definition at line 1057 of file types.f90.

**7.317.2.7 INTEGER(INTG) TYPES::VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP\_-  
TYPE::VARIABLE\_TYPE**

The variable type for this variable to equations matrices map.

Definition at line 1058 of file types.f90.

## 7.318 TYPES::VARIABLE\_TO\_SOLVER\_COL\_MAP\_TYPE Struct Reference

Contains information on the mappings between field variable dofs in an equation set and the solver matrix columns (solver dofs).

### Public Attributes

- INTEGER(INTG), allocatable [COLUMN\\_NUMBERS](#)

*COLUMN\_NUMBERS(variable\_dof\_idx)*. The solver column number (solver dof) that the variable\_dof\_idx'th variable dof is mapped to.

- REAL(DP), allocatable [COUPLING\\_COEFFICIENTS](#)

*COUPLING\_COEFFICIENTS(variable\_dof\_idx)*. The multiplicative constant for the mapping between the variable\_dof\_idx'th variable dof and the solver dof.

- REAL(DP), allocatable [ADDITIVE\\_CONSTANTS](#)

*ADDITIVE\_CONSTANTS(variable\_dof\_idx)*. The additive constant for the mapping between the variable\_dof\_idx'th variable dof and the solver dof.

### 7.318.1 Detailed Description

Contains information on the mappings between field variable dofs in an equation set and the solver matrix columns (solver dofs).

#### [Todo](#)

rename solver col to be solver dof here

Definition at line 1504 of file types.f90.

### 7.318.2 Member Data Documentation

#### 7.318.2.1 REAL(DP),allocatable TYPES::VARIABLE\_TO\_SOLVER\_COL\_MAP\_TYPE::ADDITIVE\_CONSTANTS

*ADDITIVE\_CONSTANTS(variable\_dof\_idx)*. The additive constant for the mapping between the variable\_dof\_idx'th variable dof and the solver dof.

Definition at line 1507 of file types.f90.

#### 7.318.2.2 INTEGER(INTG),allocatable TYPES::VARIABLE\_TO\_SOLVER\_COL\_MAP\_TYPE::COLUMN\_NUMBERS

*COLUMN\_NUMBERS(variable\_dof\_idx)*. The solver column number (solver dof) that the variable\_dof\_idx'th variable dof is mapped to.

Definition at line 1505 of file types.f90.

**7.318.2.3 REAL(DP),allocatable TYPES::VARIABLE\_TO\_SOLVER\_COL\_MAP\_-  
TYPE::COUPLING\_COEFFICIENTS**

COUPLING\_COEFFICIENTS(variable\_dof\_idx). The multiplicative constant for the mapping between the variable\_dof\_idx'th variable dof and the solver dof.

Definition at line 1506 of file types.f90.

## 7.319 TYPES::VECTOR\_TYPE Struct Reference

Contains information for a vector.

### Public Attributes

- INTEGER(INTG) **ID**  
*The ID of the vector.*
- LOGICAL **VECTOR\_FINISHED**  
*Is .TRUE. if the vector has finished being created, .FALSE. if not.*
- INTEGER(INTG) **N**  
*The length of the vector.*
- INTEGER(INTG) **DATA\_TYPE**  
*The data type of the vector.*
- INTEGER(INTG) **SIZE**  
*The size of the data array of the vector.*
- INTEGER(INTG), allocatable **DATA\_INTG**  
*DATA\_INTG(i). The integer data for an integer vector. The i'th component contains the data for the i'th component vector data on the domain.*
- REAL(SP), allocatable **DATA\_SP**  
*DATA\_SP(i). The real data for a single precision real vector. The i'th component contains the data for the i'th component vector data on the domain.*
- REAL(DP), allocatable **DATA\_DP**  
*DATA\_DP(i). The real data for a double precision real vector. The i'th component contains the data for the i'th component vector data on the domain.*
- LOGICAL, allocatable **DATA\_L**  
*DATA\_L(i). The real for a logical vector. The i'th component contains the data for the i'th component vector data on the domain.*

### 7.319.1 Detailed Description

Contains information for a vector.

Definition at line 569 of file types.f90.

### 7.319.2 Member Data Documentation

#### 7.319.2.1 REAL(DP),allocatable TYPES::VECTOR\_TYPE::DATA\_DP

DATA\_DP(i). The real data for a double precision real vector. The i'th component contains the data for the i'th component vector data on the domain.

Definition at line 577 of file types.f90.

**7.319.2.2 INTEGER(INTG),allocatable TYPES::VECTOR\_TYPE::DATA\_INTG**

DATA\_INTG(i). The integer data for an integer vector. The i'th component contains the data for the i'th component vector data on the domain.

Definition at line 575 of file types.f90.

**7.319.2.3 LOGICAL,allocatable TYPES::VECTOR\_TYPE::DATA\_L**

DATA\_L(i). The real for a logical vector. The i'th component contains the data for the i'th component vector data on the domain.

Definition at line 578 of file types.f90.

**7.319.2.4 REAL(SP),allocatable TYPES::VECTOR\_TYPE::DATA\_SP**

DATA\_SP(i). The real data for a single precision real vector. The i'th component contains the data for the i'th component vector data on the domain.

Definition at line 576 of file types.f90.

**7.319.2.5 INTEGER(INTG) TYPES::VECTOR\_TYPE::DATA\_TYPE**

The data type of the vector.

**See also:**

[MATRIX\\_VECTOR::DataTypes](#)

Definition at line 573 of file types.f90.

**7.319.2.6 INTEGER(INTG) TYPES::VECTOR\_TYPE::ID**

The ID of the vector.

Definition at line 570 of file types.f90.

**7.319.2.7 INTEGER(INTG) TYPES::VECTOR\_TYPE::N**

The length of the vector.

Definition at line 572 of file types.f90.

**7.319.2.8 INTEGER(INTG) TYPES::VECTOR\_TYPE::SIZE**

The size of the data array of the vector.

Definition at line 574 of file types.f90.

**7.319.2.9 LOGICAL TYPES::VECTOR\_TYPE::VECTOR\_FINISHED**

Is .TRUE. if the vector has finished being created, .FALSE. if not.

Definition at line 571 of file types.f90.



## **Chapter 8**

# **File Documentation**

### **8.1 additional\_doc/Installation.dox File Reference**

## 8.2 additional\_doc/library\_commands.dox File Reference

How to add support for a programming language.

### 8.2.1 Detailed Description

How to add support for a programming language.

Definition in file [library\\_commands.dox](#).

## 8.3 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/analytic\_analysis\_routines.f90 File Reference

### Id

[analytic\\_analysis\\_routines.f90](#) 178 2008-10-26 13:07:57Z chrisbradley

### Namespaces

- namespace [ANALYTIC\\_ANALYSIS\\_ROUTINES](#)

*This module handles all analytic analysis routines.*

### Functions

- subroutine [ANALYTIC\\_ANALYSIS\\_ROUTINES::ANALYTIC\\_ANALYSIS\\_CALCULATE](#) (FIELD, FILE\_ID, ERR, ERROR,\*)

*Calculate the analytic analysis data.*

- subroutine [ANALYTIC\\_ANALYSIS\\_ROUTINES::ANALYTIC\\_ANALYSIS\\_EXPORT](#) (FIELD, FILE\_NAME, METHOD, ERR, ERROR,\*)

*Export analytic information.*

- subroutine [ANALYTIC\\_ANALYSIS\\_ROUTINES::ANALYTIC\\_ANALYSIS\\_FORTRAN\\_FILE\\_OPEN](#) (FILE\_ID, FILE\_NAME, FILE\_STATUS, ERR, ERROR,\*)

*Open a file using Fortran. TODO should we use method in FIELD\_IO??*

- subroutine [ANALYTIC\\_ANALYSIS\\_ROUTINES::ANALYTIC\\_ANALYSIS\\_FORTRAN\\_FILE\\_WRITE\\_STRING](#) (FILE\_ID, STRING\_DATA, LEN\_OF\_DATA, ERR, ERROR,\*)

*Write a string using FORTRAN IO. TODO should we use FIELD\_IO\_FORTRAN\_FILE\_WRITE\_STRING??*

- subroutine [ANALYTIC\\_ANALYSIS\\_ROUTINES::ANALYTIC\\_ANALYSIS\\_INTEGRAL\\_ABSOLUTE\\_ERROR\\_GET](#) (FIELD, VARIABLE\_NUMBER, POWER, VALUE, ERR, ERROR,\*)

*Get integral absolute error value for the field.*

- subroutine [ANALYTIC\\_ANALYSIS\\_ROUTINES::ANALYTIC\\_ANALYSIS\\_INTEGRAL\\_ANALYTIC\\_VALUE\\_GET](#) (FIELD, VARIABLE\_NUMBER, POWER, VALUE, ERR, ERROR,\*)

*Get integral analytic value for the field TODO should we use analytical formula to calculate the integration?*

- subroutine [ANALYTIC\\_ANALYSIS\\_ROUTINES::ANALYTIC\\_ANALYSIS\\_INTEGRAL\\_NUMERICAL\\_VALUE\\_GET](#) (FIELD, VARIABLE\_NUMBER, POWER, VALUE, ERR, ERROR,\*)

*Get integral numerical value for the field, TODO check integral calculation.*

- subroutine [ANALYTIC\\_ANALYSIS\\_ROUTINES::ANALYTIC\\_ANALYSIS\\_INTEGRAL\\_PERCENT\\_ERROR\\_GET](#) (FIELD, VARIABLE\_NUMBER, POWER, VALUE, ERR, ERROR,\*)

*Get integral percentage error value for the field.*

- subroutine [ANALYTIC\\_ANALYSIS\\_ROUTINES::ANALYTIC\\_ANALYSIS\\_INTEGRAL\\_RELATIVE\\_ERROR\\_GET](#) (FIELD, VARIABLE\_NUMBER, POWER, VALUE, ERR, ERROR,\*)
 

*Get integral relative error value for the field.*
- subroutine [ANALYTIC\\_ANALYSIS\\_ROUTINES::ANALYTIC\\_ANALYSIS\\_NODE\\_ABSOLUTE\\_ERROR\\_GET](#) (FIELD, VARIABLE\_NUMBER, COMPONENT\_NUMBER, NODE\_NUMBER, DERIVATIVE\_NUMBER, VALUE, ERR, ERROR,\*)
 

*Get absolute error value for the node.*
- subroutine [ANALYTIC\\_ANALYSIS\\_ROUTINES::ANALYTIC\\_ANALYSIS\\_NODE\\_ANALYTIC\\_VALUE\\_GET](#) (FIELD, VARIABLE\_NUMBER, COMPONENT\_NUMBER, NODE\_NUMBER, DERIVATIVE\_NUMBER, VALUE, ERR, ERROR,\*)
 

*Get Analytic value for the node.*
- subroutine [ANALYTIC\\_ANALYSIS\\_ROUTINES::ANALYTIC\\_ANALYSIS\\_NODE\\_NUMERICAL\\_VALUE\\_GET](#) (FIELD, VARIABLE\_NUMBER, COMPONENT\_NUMBER, NODE\_NUMBER, DERIVATIVE\_NUMBER, VALUE, ERR, ERROR,\*)
 

*Get Numerical value for the node.*
- subroutine [ANALYTIC\\_ANALYSIS\\_ROUTINES::ANALYTIC\\_ANALYSIS\\_NODE\\_PERCENT\\_ERROR\\_GET](#) (FIELD, VARIABLE\_NUMBER, COMPONENT\_NUMBER, NODE\_NUMBER, DERIVATIVE\_NUMBER, VALUE, ERR, ERROR,\*)
 

*Get percentage error value for the node.*
- subroutine [ANALYTIC\\_ANALYSIS\\_ROUTINES::ANALYTIC\\_ANALYSIS\\_NODE\\_RELATIVE\\_ERROR\\_GET](#) (FIELD, VARIABLE\_NUMBER, COMPONENT\_NUMBER, NODE\_NUMBER, DERIVATIVE\_NUMBER, VALUE, ERR, ERROR,\*)
 

*Get relative error value for the node.*
- subroutine [ANALYTIC\\_ANALYSIS\\_ROUTINES::ANALYTIC\\_ANALYSIS\\_RMS\\_PERCENT\\_ERROR\\_GET](#) (FIELD, VARIABLE\_NUMBER, VALUE, ERR, ERROR,\*)
 

*Get rms percentage error value for the field.*
- subroutine [ANALYTIC\\_ANALYSIS\\_ROUTINES::ANALYTIC\\_ANALYSIS\\_RMS\\_ABSOLUTE\\_ERROR\\_GET](#) (FIELD, VARIABLE\_NUMBER, VALUE, ERR, ERROR,\*)
 

*Get rms absolute error value for the field.*
- subroutine [ANALYTIC\\_ANALYSIS\\_ROUTINES::ANALYTIC\\_ANALYSIS\\_RMS\\_RELATIVE\\_ERROR\\_GET](#) (FIELD, VARIABLE\_NUMBER, VALUE, ERR, ERROR,\*)
 

*Get rms relative error value for the field.*

### 8.3.1 Detailed Description

#### Id

[analytic\\_analysis\\_routines.f90](#) 178 2008-10-26 13:07:57Z chrispbradley

#### Author:

Ting Yu This module handles all analytic analysis routines.

### **8.3.2 LICENSE**

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [analytic\\_analysis\\_routines.f90](#).

## 8.4 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/base\_routines.f90 File Reference

### Id

[base\\_routines.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

### Classes

- struct [BASE\\_ROUTINES::ROUTINE\\_LIST\\_ITEM\\_TYPE](#)  
*Contains information for an item in the routine list for diagnostics or timing.*
- struct [BASE\\_ROUTINES::ROUTINE\\_LIST\\_TYPE](#)  
*Contains information for the routine list for diagnostics or timing.*
- struct [BASE\\_ROUTINES::ROUTINE\\_STACK\\_ITEM\\_TYPE](#)  
*Contains information for an item in the routine invocation stack.*
- struct [BASE\\_ROUTINES::ROUTINE\\_STACK\\_TYPE](#)  
*Contains information for the routine invocation stack.*
- interface [BASE\\_ROUTINES::interface](#)
- interface [BASE\\_ROUTINES::FLAG\\_ERROR](#)  
*Flags an error condition.*
- interface [BASE\\_ROUTINES::FLAG\\_WARNING](#)  
*Flags a warning to the user.*

### Namespaces

- namespace [BASE\\_ROUTINES](#)  
*This module contains all the low-level base routines e.g., all debug, control, and low-level communication routines.*

### Functions

- subroutine [BASE\\_ROUTINES::ENTERS](#) (NAME, ERR, ERROR,\*)  
*Records the entry into the named procedure and initialises the error code.*
- subroutine [BASE\\_ROUTINES::ERRORS](#) (NAME, ERR, ERROR)  
*Records the exiting error of the subroutine.*
- subroutine [BASE\\_ROUTINES::EXITS](#) (NAME)  
*Records the exit out of the named procedure.*
- subroutine [BASE\\_ROUTINES::FLAG\\_ERROR\\_C](#) (STRING, ERR, ERROR,\*)  
*Sets the error string specified by a character string and flags an error.*

- subroutine **BASE\_ROUTINES::FLAG\_ERROR\_VS** (STRING, ERR, ERROR,\*)  
*Sets the error string specified by a varying string and flags an error.*
- subroutine **BASE\_ROUTINES::FLAG\_WARNING\_C** (STRING, ERR, ERROR,\*)  
*Writes a warning message specified by a character string to the user.*
- subroutine **BASE\_ROUTINES::FLAG\_WARNING\_VS** (STRING, ERR, ERROR,\*)  
*Writes a warning message specified by a varying string to the user.*
- subroutine **BASE\_ROUTINES::BASE\_ROUTINES\_FINALISE** (ERR, ERROR,\*)  
*Finalises the base\_routines module and deallocates all memory.*
- subroutine **BASE\_ROUTINES::BASE\_ROUTINES\_INITIALISE** (ERR, ERROR,\*)  
*Initialises the variables required for the base\_routines module.*
- subroutine **BASE\_ROUTINES::DIAGNOSTICS\_SET\_OFF** (ERR, ERROR,\*)  
*Sets diagnostics off.*
- subroutine **BASE\_ROUTINES::DIAGNOSTICS\_SET\_ON** (DIAG\_TYPE, LEVEL\_LIST, DIAG\_FILENAME, ROUTINE\_LIST, ERR, ERROR,\*)  
*Sets diagnostics on.*
- subroutine **BASE\_ROUTINES::OUTPUT\_SET\_OFF** (ERR, ERROR,\*)  
*Sets writes file echo output off.*
- subroutine **BASE\_ROUTINES::OUTPUT\_SET\_ON** (ECHO\_FILENAME, ERR, ERROR,\*)  
*Sets writes file echo output on.*
- subroutine **BASE\_ROUTINES::TIMING\_SET\_OFF** (ERR, ERROR,\*)  
*Sets timing off.*
- subroutine **BASE\_ROUTINES::TIMING\_SET\_ON** (TIMING\_TYPE, TIMING\_SUMMARY\_FLAG, TIMING\_FILENAME, ROUTINE\_LIST, ERR, ERROR,\*)  
*Sets timing on.*
- subroutine **BASE\_ROUTINES::TIMING\_SUMMARY\_OUTPUT** (ERR, ERROR,\*)  
*Outputs the timing summary.*
- subroutine **BASE\_ROUTINES::WRITE\_STR** (ID, ERR, ERROR,\*)  
*Writes the output string to a specified output stream.*

## Variables

- INTEGER(INTG), parameter **BASE\_ROUTINES::MAX\_OUTPUT\_LINES** = 500  
*Maximum number of lines that can be output.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::GENERAL\_OUTPUT\_TYPE** = 1

*General output type.*

- INTEGER(INTG), parameter **BASE\_ROUTINES::DIAGNOSTIC\_OUTPUT\_TYPE** = 2  
*Diagnostic output type.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::TIMING\_OUTPUT\_TYPE** = 3  
*Timing output type.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::ERROR\_OUTPUT\_TYPE** = 4  
*Error output type.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::HELP\_OUTPUT\_TYPE** = 5  
*Help output type.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::ECHO\_FILE\_UNIT** = 10  
*File unit for echo files.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::DIAGNOSTICS\_FILE\_UNIT** = 11  
*File unit for diagnostic files.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::TIMING\_FILE\_UNIT** = 12  
*File unit for timing files.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::LEARN\_FILE\_UNIT** = 13  
*File unit for learn files.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::IO1\_FILE\_UNIT** = 21  
*File unit for general IO 1 files.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::IO2\_FILE\_UNIT** = 22  
*File unit for general IO 2 files.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::IO3\_FILE\_UNIT** = 23  
*File unit for general IO 3 files.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::IO4\_FILE\_UNIT** = 24  
*File unit for general IO 4 files.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::IO5\_FILE\_UNIT** = 25  
*File unit for general IO 5 files.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::TEMPORARY\_FILE\_UNIT** = 80  
*File unit for temporary files.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::OPEN\_COMFILE\_UNIT** = 90  
*File unit for open command files.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::START\_READ\_COMFILE\_UNIT** = 90  
*First file unit for read command files.*

- INTEGER(INTG), parameter **BASE\_ROUTINES::STOP\_READ\_COMFILE\_UNIT** = 99  
*Last file unit for read command files.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::ALL\_DIAG\_TYPE** = 1  
*Type for setting diagnostic output in all routines.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::IN\_DIAG\_TYPE** = 2  
*Type for setting diagnostic output in one routine.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::FROM\_DIAG\_TYPE** = 3  
*Type for setting diagnostic output from one routine downwards.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::ALL\_TIMING\_TYPE** = 1  
*Type for setting timing output in all routines.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::IN\_TIMING\_TYPE** = 2  
*Type for setting timing output in one routine.*
- INTEGER(INTG), parameter **BASE\_ROUTINES::FROM\_TIMING\_TYPE** = 3  
*Type for setting timing output from one routine downwards.*
- LOGICAL, save **BASE\_ROUTINES::DIAGNOSTICS**  
*.TRUE. if diagnostic output is required in any routines.*
- LOGICAL, save **BASE\_ROUTINES::DIAGNOSTICS1**  
*.TRUE. if level 1 diagnostic output is active in the current routine*
- LOGICAL, save **BASE\_ROUTINES::DIAGNOSTICS2**  
*.TRUE. if level 2 diagnostic output is active in the current routine*
- LOGICAL, save **BASE\_ROUTINES::DIAGNOSTICS3**  
*.TRUE. if level 3 diagnostic output is active in the current routine*
- LOGICAL, save **BASE\_ROUTINES::DIAGNOSTICS4**  
*.TRUE. if level 4 diagnostic output is active in the current routine*
- LOGICAL, save **BASE\_ROUTINES::DIAGNOSTICS5**  
*.TRUE. if level 5 diagnostic output is active in the current routine*
- LOGICAL, save **BASE\_ROUTINES::DIAGNOSTICS\_LEVEL1**  
*.TRUE. if the user has requested level 1 diagnostic output to be active*
- LOGICAL, save **BASE\_ROUTINES::DIAGNOSTICS\_LEVEL2**  
*.TRUE. if the user has requested level 2 diagnostic output to be active*
- LOGICAL, save **BASE\_ROUTINES::DIAGNOSTICS\_LEVEL3**  
*.TRUE. if the user has requested level 3 diagnostic output to be active*
- LOGICAL, save **BASE\_ROUTINES::DIAGNOSTICS\_LEVEL4**  
*.TRUE. if the user has requested level 4 diagnostic output to be active*

- LOGICAL, save **BASE\_ROUTINES::DIAGNOSTICS\_LEVELS**  
*.TRUE. if the user has requested level 5 diagnostic output to be active*
- LOGICAL, save **BASE\_ROUTINES::DIAG\_ALL\_SUBROUTINES**  
*.TRUE. if diagnostic output is required in all routines*
- LOGICAL, save **BASE\_ROUTINES::DIAG\_FROM\_SUBROUTINE**  
*.TRUE. if diagnostic output is required from a particular routine*
- LOGICAL, save **BASE\_ROUTINES::DIAG\_FILE\_OPEN**  
*.TRUE. if the diagnostic output file is open*
- LOGICAL, save **BASE\_ROUTINES::DIAG\_OR\_TIMING**  
*.TRUE. if diagnostics or time is .TRUE.*
- LOGICAL, save **BASE\_ROUTINES::ECHO\_OUTPUT**  
*.TRUE. if all output is to be echoed to the echo file*
- LOGICAL, save **BASE\_ROUTINES::TIMING**  
*.TRUE. if timing output is required in any routines.*
- LOGICAL, save **BASE\_ROUTINES::TIMING\_SUMMARY**  
*.TRUE. if timing output will be summary form via a TIMING\_SUMMARY\_OUTPUT call otherwise timing will be output for routines when the routine exits*
- LOGICAL, save **BASE\_ROUTINES::TIMING\_ALL\_SUBROUTINES**  
*.TRUE. if timing output is required in all routines*
- LOGICAL, save **BASE\_ROUTINES::TIMING\_FROM\_SUBROUTINE**  
*.TRUE. if timing output is required from a particular routine*
- LOGICAL, save **BASE\_ROUTINES::TIMING\_FILE\_OPEN**  
*.TRUE. if the timing output file is open*
- CHARACTER(LEN=MAXSTRLEN), save **BASE\_ROUTINES::OP\_STRING**  
*The array of lines to output.*
- TYPE(ROUTINE\_LIST\_TYPE), save **BASE\_ROUTINES::DIAG\_ROUTINE\_LIST**  
*The list of routines for which diagnostic output is required.*
- TYPE(ROUTINE\_LIST\_TYPE), save **BASE\_ROUTINES::TIMING\_ROUTINE\_LIST**  
*The list of routines for which timing output is required.*
- TYPE(ROUTINE\_STACK\_TYPE), save **BASE\_ROUTINES::ROUTINE\_STACK**  
*The routime invocation stack.*

### 8.4.1 Detailed Description

#### Id

[base\\_routines.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

#### Author:

Chris Bradley This module contains all the low-level base routines e.g., all debug, control, and low-level communication routines.

### 8.4.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

#### Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [base\\_routines.f90](#).

## 8.5 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/basis\_-routines.f90 File Reference

## 8.6 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/binary\_file\_c.c File Reference

Include dependency graph for binary\_file\_c.c:

### Defines

- #define INTEGERTYPE 1
- #define SHORTINTTYPE 2
- #define LONGINTTYPE 3
- #define FLOATTYPE 4
- #define DOUBLETYPE 5
- #define QUADRUPLETYP 6
- #define CHARTYPE 7
- #define LOGICALTYPE 8
- #define COMPLEXTYPE 9
- #define DOUBLECOMPLEX 10
- #define QUADRUPLECOMPLEX 11
- #define MAXBINFILES 99
- #define SAMEENDIAN 0
- #define FLIPENDIAN 1

### Typedefs

- typedef int logical

### Functions

- void BinaryCloseFile (int \*fileid, int \*err, char \*error\_string)
- void BinaryOpenFile (int \*fileid, char \*filename, char \*access\_code, int \*err, char \*error\_string)
- void BinaryReadFile (int \*fileid, int \*endian, int \*number\_of\_items, int \*item\_type, char \*data, int \*err, char \*error\_string)
- void BinarySetFile (int \*fileid, int \*set\_code, int \*err, char \*error\_string)
- void BinarySkipFile (int \*fileid, int \*number\_of\_bytes, int \*err, char \*error\_string)
- void BinaryWriteFile (int \*fileid, int \*endian, int \*number\_of\_items, int \*item\_type, char \*data, int \*err, char \*error\_string)
- void IsBinaryFileOpen (int \*fileid, int \*returncode, int \*err, char \*error\_string)
- void IsEndBinaryFile (int \*fileid, int \*returncode, int \*err, char \*error\_string)

### Variables

- FILE \* binaryfiles [MAXBINFILES]

#### 8.6.1 Define Documentation

##### 8.6.1.1 #define CHARTYPE 7

Definition at line 118 of file binary\_file\_c.c.

Referenced by BinaryReadFile(), and BinaryWriteFile().

**8.6.1.2 #define COMPLEXTYPE 9**

Definition at line 120 of file binary\_file\_c.c.

**8.6.1.3 #define DOUBLECOMPLEXTYPE 10**

Definition at line 121 of file binary\_file\_c.c.

**8.6.1.4 #define DOUBLETYPE 5**

Definition at line 116 of file binary\_file\_c.c.

Referenced by BinaryReadFile(), and BinaryWriteFile().

**8.6.1.5 #define FLIPENDIAN 1**

Definition at line 128 of file binary\_file\_c.c.

**8.6.1.6 #define FLOATTYPE 4**

Definition at line 115 of file binary\_file\_c.c.

Referenced by BinaryReadFile(), and BinaryWriteFile().

**8.6.1.7 #define INTEGERTYPE 1**

Definition at line 112 of file binary\_file\_c.c.

Referenced by BinaryReadFile(), and BinaryWriteFile().

**8.6.1.8 #define LOGICALTYPE 8**

Definition at line 119 of file binary\_file\_c.c.

Referenced by BinaryReadFile(), and BinaryWriteFile().

**8.6.1.9 #define LONGINTTYPE 3**

Definition at line 114 of file binary\_file\_c.c.

**8.6.1.10 #define MAXBINFILES 99**

Definition at line 126 of file binary\_file\_c.c.

Referenced by BinaryCloseFile(), BinaryOpenFile(), BinaryReadFile(), BinarySetFile(), BinarySkipFile(), BinaryWriteFile(), IsBinaryFileOpen(), and IsEndBinaryFile().

**8.6.1.11 #define QUADRUPLECOMPLEXTYPE 11**

Definition at line 122 of file binary\_file\_c.c.

### 8.6.1.12 #define QUADRUPLETTYPE 6

Definition at line 117 of file binary\_file\_c.c.

### 8.6.1.13 #define SAMEENDIAN 0

Definition at line 127 of file binary\_file\_c.c.

Referenced by BinaryReadFile(), and BinaryWriteFile().

### 8.6.1.14 #define SHORTINTTYPE 2

Definition at line 113 of file binary\_file\_c.c.

Referenced by BinaryReadFile(), and BinaryWriteFile().

## 8.6.2 Typedef Documentation

### 8.6.2.1 typedef int logical

Definition at line 132 of file binary\_file\_c.c.

## 8.6.3 Function Documentation

### 8.6.3.1 void BinaryCloseFile (int \*fileid, int \*err, char \*error\_string)

Definition at line 195 of file binary\_file\_c.c.

References binaryfiles, and MAXBINFILES.

### 8.6.3.2 void BinaryOpenFile (int \*fileid, char \*filename, char \*access\_code, int \*err, char \*error\_string)

Definition at line 227 of file binary\_file\_c.c.

References binaryfiles, and MAXBINFILES.

### 8.6.3.3 void BinaryReadFile (int \*fileid, int \*endian, int \*number\_of\_items, int \*item\_type, char \*data, int \*err, char \*error\_string)

Definition at line 265 of file binary\_file\_c.c.

References binaryfiles, CHARTYPE, DOUBLETYPE, FLOATTYPE, INTEGERTYPE, LOGICALTYPE, MAXBINFILES, SAMEENDIAN, and SHORTINTTYPE.

### 8.6.3.4 void BinarySetFile (int \*fileid, int \*set\_code, int \*err, char \*error\_string)

Definition at line 375 of file binary\_file\_c.c.

References binaryfiles, and MAXBINFILES.

### **8.6.3.5 void BinarySkipFile (int \*fileid, int \*number\_of\_bytes, int \*err, char \*error\_string)**

Definition at line 440 of file binary\_file\_c.c.

References binaryfiles, and MAXBINFILES.

### **8.6.3.6 void BinaryWriteFile (int \*fileid, int \*endian, int \*number\_of\_items, int \*item\_type, char \*data, int \*err, char \*error\_string)**

Definition at line 483 of file binary\_file\_c.c.

References binaryfiles, CHARTYPE, DOUBLETYPE, FLOATTYPE, INTEGERTYPE, LOGICALTYPE, MAXBINFILES, SAMEENDIAN, and SHORTINTTYPE.

### **8.6.3.7 void IsBinaryFileOpen (int \*fileid, int \*returncode, int \*err, char \*error\_string)**

Definition at line 587 of file binary\_file\_c.c.

References binaryfiles, and MAXBINFILES.

### **8.6.3.8 void IsEndBinaryFile (int \*fileid, int \*returncode, int \*err, char \*error\_string)**

Definition at line 621 of file binary\_file\_c.c.

References binaryfiles, and MAXBINFILES.

## **8.6.4 Variable Documentation**

### **8.6.4.1 FILE\* binaryfiles[MAXBINFILES]**

#### **Initial value:**

```
{NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL}
```

Definition at line 180 of file binary\_file\_c.c.

Referenced by BinaryCloseFile(), BinaryOpenFile(), BinaryReadFile(), BinarySetFile(), BinarySkipFile(), BinaryWriteFile(), IsBinaryFileOpen(), and IsEndBinaryFile().

## 8.7 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/binary\_file\_f.f90 File Reference

### Id

[binary\\_file\\_f.f90](#) 177 2008-10-25 13:38:18Z chrispb Bradley

### Classes

- struct [BINARY\\_FILE::BINARY\\_FILE\\_INFO\\_TYPE](#)
- struct [BINARY\\_FILE::BINARY\\_FILE\\_TYPE](#)
- struct [BINARY\\_FILE::BINARY\\_TAG\\_TYPE](#)
- interface [BINARY\\_FILE::interface](#)
- interface [BINARY\\_FILE::READ\\_BINARY\\_FILE](#)
- interface [BINARY\\_FILE::WRITE\\_BINARY\\_FILE](#)

### Namespaces

- namespace [BINARY\\_FILE](#)

*This module handles the reading and writing of binary files.*

### Functions

- LOGICAL [BINARY\\_FILE::INQUIRE\\_OPEN\\_BINARY\\_FILE](#) (FILEID)
- LOGICAL [BINARY\\_FILE::INQUIRE\\_EOF\\_BINARY\\_FILE](#) (FILEID, ERR, ERROR)
- subroutine [BINARY\\_FILE::CLOSE\\_BINARY\\_FILE](#) (FILEID, ERR, ERROR,\*)
- subroutine [BINARY\\_FILE::CLOSE\\_CMISS\\_BINARY\\_FILE](#) (FILEID, ERR, ERROR,\*)
- subroutine [BINARY\\_FILE::OPEN\\_BINARY\\_FILE](#) (FILEID, COMMAND, FILENAME, ERR, ERROR,\*)
- subroutine [BINARY\\_FILE::OPEN\\_CMISS\\_BINARY\\_FILE](#) (FILEID, FILE\_TYPE, NUMBER\_TA\_S,&VERSION, FILEVERSION, COMMAND, EXTENSION, FILENAME, ERR, ERROR,\*)
- subroutine [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_INTG](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_INTG1](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_SINTG](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_SINTG1](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_LINTG](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_LINTG1](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_SP](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_SP1](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [BINARY\\_FILE::READ\\_BINARY\\_FILE\\_DP](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)

- subroutine `BINARY_FILE::READ_BINARY_FILE_DP1` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::READ_BINARY_FILE_CHARACTER` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::READ_BINARY_FILE_LOGICAL` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::READ_BINARY_FILE_LOGICAL1` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::READ_BINARY_FILE_SPC` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::READ_BINARY_FILE_SPC1` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::READ_BINARY_FILE_DPC` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::READ_BINARY_FILE_DPC1` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::READ_BINARY_TAG_HEADER` (FILEID, TAG, ERR, ERROR,\*)
- subroutine `BINARY_FILE::RESET_BINARY_NUMBER_TAGS` (FILEID, NUMBER\_TAGS, ERR, ERROR,\*)
- subroutine `BINARY_FILE::SET_BINARY_FILE` (FILEID, SET\_CODE, ERR, ERROR,\*)
- subroutine `BINARY_FILE::SKIP_CM_BINARY_HEADER` (FILEID, SKIP, ERR, ERROR,\*)
- subroutine `BINARY_FILE::SKIP_BINARY_FILE` (FILEID, NUMBER\_BYTES, ERR, ERROR,\*)
- subroutine `BINARY_FILE::SKIP_BINARY_TAGS` (FILEID, TAG, ERR, ERROR,\*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_INTG` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_INTG1` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_SINTG` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_SINTG1` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_LINTG` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_LINTG1` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_SP` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_SP1` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_DP` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_DP1` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_CHARACTER` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_LOGICAL` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_LOGICAL1` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_SPC` (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)

- subroutine [BINARY\\_FILE::WRITE\\_BINARY\\_FILE\\_SPC1](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [BINARY\\_FILE::WRITE\\_BINARY\\_FILE\\_DPC](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [BINARY\\_FILE::WRITE\\_BINARY\\_FILE\\_DPC1](#) (FILEID, NUM\_DATA, DATA, ERR, ERROR,\*)
- subroutine [BINARY\\_FILE::WRITE\\_BINARY\\_TAG\\_HEADER](#) (FILEID, TAG, ERR, ERROR,\*)

## Variables

- INTEGER(INTG), dimension, parameter [BINARY\\_FILE::MAX\\_NUM\\_BINARY\\_FILES](#) = 99
- INTEGER(INTG), parameter [BINARY\\_FILE::FILE\\_BEGINNING](#) = 0
- INTEGER(INTG), parameter [BINARY\\_FILE::FILE\\_CURRENT](#) = 1
- INTEGER(INTG), parameter [BINARY\\_FILE::FILE\\_END](#) = 2
- INTEGER(INTG), parameter [BINARY\\_FILE::FILE\\_SAME\\_ENDIAN](#) = 0
- INTEGER(INTG), parameter [BINARY\\_FILE::FILE\\_CHANGE\\_ENDIAN](#) = 1
- INTEGER(INTG), parameter [BINARY\\_FILE::BINARY\\_FILE\\_READABLE](#) = 1
- INTEGER(INTG), parameter [BINARY\\_FILE::BINARY\\_FILE\\_WRITABLE](#) = 2
- INTEGER(INTG), parameter [BINARY\\_FILE::CMISS\\_BINARY\\_IDENTITY](#) = 7
- INTEGER(INTG), parameter [BINARY\\_FILE::CMISS\\_BINARY\\_MATRIX\\_FILE](#) = 1
- INTEGER(INTG), parameter [BINARY\\_FILE::CMISS\\_BINARY\\_HISTORY\\_FILE](#) = 2
- INTEGER(INTG), parameter [BINARY\\_FILE::CMISS\\_BINARY\\_SIGNAL\\_FILE](#) = 3
- INTEGER(INTG), parameter [BINARY\\_FILE::CMISS\\_BINARY\\_IDENTITY\\_HEADER](#) = 1
- INTEGER(INTG), parameter [BINARY\\_FILE::CMISS\\_BINARY\\_MACHINE\\_HEADER](#) = 2
- INTEGER(INTG), parameter [BINARY\\_FILE::CMISS\\_BINARY\\_FILE\\_HEADER](#) = 3
- LOGICAL, save [BINARY\\_FILE::BINARY\\_FILE\\_USED](#) = .FALSE.

### 8.7.1 Detailed Description

#### Id

[binary\\_file\\_f.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

#### Author:

Chris Bradley This module handles the reading and writing of binary files.

#### Todo

Fix naming convention and update to current code standard.

### 8.7.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [binary\\_file\\_f.f90](#).

## 8.8 d:/Users/tyu011/workspace/OpenCMISS-trunk/srcblas.f90 File Reference

### Id

[blas.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

### Classes

- interface [BLAS::interface](#)

### Namespaces

- namespace [BLAS](#)

*This module contains the interface descriptions to the BLAS routines.*

#### 8.8.1 Detailed Description

### Id

[blas.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

### Author:

Chris Bradley This module contains the interface descriptions to the [BLAS](#) routines.

#### 8.8.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL

or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [blas.f90](#).

## 8.9 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/classical\_field\_routines.f90 File Reference

### Id

[classical\\_field\\_routines.f90](#) 180 2008-10-26 16:45:43Z chrispb  
chrispb

### Namespaces

- namespace [CLASSICAL\\_FIELD\\_ROUTINES](#)  
*This module handles all classical field class routines.*

### Functions

- subroutine [CLASSICAL\\_FIELD\\_ROUTINES::CL](#) (EQUATIONS\_SET, EQUATIONS\_TYPE, EQUATIONS\_SUBTYPE,&ERR, ERROR,\*)  
*Gets the problem type and subtype for a classical field equation set class.*
- subroutine [CLASSICAL\\_FIELD\\_ROUTINES::CL](#) (EQUATIONS\_SET, EQUATIONS\_TYPE, EQUATIONS\_SUBTYPE,&ERR, ERROR,\*)  
*Sets/changes the problem type and subtype for a classical field equation set class.*
- subroutine [CLASSICAL\\_FIELD\\_ROUTINES::CLASSICAL\\_FIELDFINITEELEMENTCALCULATE](#) (EQUATIONS\_SET, ELEMENT\_NUMBER, ERR, ERROR,\*)  
*Calculates the element stiffness matrices and rhs vector for the given element number for a classical field class finite element equation set.*
- subroutine [CLASSICAL\\_FIELD\\_ROUTINES::CLASSICAL\\_FIELDFINITEELEMENTJACOBIAN\\_EVALUATE](#) (EQUATIONS\_SET, ELEMENT\_NUMBER, ERR, ERROR,\*)  
*Evaluates the element Jacobian matrix for the given element number for a classical field class finite element equation set.*
- subroutine [CLASSICAL\\_FIELD\\_ROUTINES::CLASSICAL\\_FIELDFINITEELEMENTRESIDUAL\\_EVALUATE](#) (EQUATIONS\_SET, ELEMENT\_NUMBER, ERR, ERROR,\*)  
*Evaluates the element residual and rhs vectors for the given element number for a classical field class finite element equation set.*
- subroutine [CLASSICAL\\_FIELD\\_ROUTINES::CLASSICAL\\_FIELD\\_EQUATIONS\\_SET\\_SETUP](#) (EQUATIONS\_SET, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)  
*Sets up the equations set for a classical field equations set class.*
- subroutine [CLASSICAL\\_FIELD\\_ROUTINES::CLASSICAL\\_FIELD\\_PROBLEM\\_CLASS\\_TYPE\\_GET](#) (PROBLEM, PROBLEM\_EQUATION\_TYPE, PROBLEM\_SUBTYPE, ERR, ERROR,\*)  
*Gets the problem type and subtype for a classical field problem class.*
- subroutine [CLASSICAL\\_FIELD\\_ROUTINES::CLASSICAL\\_FIELD\\_PROBLEM\\_CLASS\\_TYPE\\_SET](#) (PROBLEM, PROBLEM\_EQUATION\_TYPE, PROBLEM\_SUBTYPE, ERR, ERROR,\*)

*Sets/changes the problem type and subtype for a classical field problem class.*

- subroutine      **CLASSICAL\_FIELD\_ROUTINES::CLASSICAL\_FIELD\_PROBLEM\_SETUP**  
(PROBLEM, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)

*Sets up the problem for a classical field problem class.*

### 8.9.1 Detailed Description

#### Id

[classical\\_field\\_routines.f90](#) 180 2008-10-26 16:45:43Z chrispbradley

#### Author:

Chris Bradley This module handles all classical field routines.

### 8.9.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [classical\\_field\\_routines.f90](#).

## 8.10 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/cmiss.f90 File Reference

### Id

[cmiss.f90](#) 178 2008-10-26 13:07:57Z chrispbradley

### Namespaces

- namespace [CMISS](#)  
*The top level cmiss module.*

### Functions

- subroutine [CMISS::CMISS\\_FINALISE](#) (ERR, ERROR,\*)  
*Finalises [CMISS](#).*
- subroutine [CMISS::CMISS\\_INITIALISE](#) (ERR, ERROR,\*)  
*Initialises [CMISS](#).*
- subroutine [CMISS::CMISS\\_WRITE\\_ERROR](#) (ERR, ERROR)  
*Writes the error string to screen.*

### Variables

- INTEGER(INTG), parameter [CMISS::CMISS\\_MAJOR\\_VERSION](#) = 0
- INTEGER(INTG), parameter [CMISS::CMISS\\_MINOR\\_VERSION](#) = 2
- INTEGER(INTG), parameter [CMISS::CMISS\\_REVISION\\_VERSION](#) = 0
- CHARACTER(LEN=MAXSTRLEN), parameter [CMISS::CMISS\\_BUILD\\_VERSION](#)

#### 8.10.1 Detailed Description

### Id

[cmiss.f90](#) 178 2008-10-26 13:07:57Z chrispbradley

### Author:

Chris Bradley The top level cmiss module.

Definition in file [cmiss.f90](#).

## 8.11 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/cmiss\_-mpi.f90 File Reference

### Id

cmiss\_mpi.f90 177 2008-10-25 13:38:18Z chrisbradley

### Namespaces

- namespace **CMISS\_MPI**

*This module contains CMISS MPI routines.*

### Functions

- subroutine **CMISS\_MPI::MPI\_ERROR\_CHECK** (ROUTINE, MPI\_ERR\_CODE, ERR, ERROR,\*)

*Checks to see if an MPI error has occurred during an MPI call and flags a CMISS error if it has.*

#### 8.11.1 Detailed Description

### Id

cmiss\_mpi.f90 177 2008-10-25 13:38:18Z chrisbradley

### Author:

Chris Bradley This module contains CMISS MPI routines.

#### 8.11.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and

not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [cmiss\\_mpi.f90](#).

## 8.12 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/cmiss\_-parmetis.f90 File Reference

### Id

[cmiss\\_parmetis.f90](#) 177 2008-10-25 13:38:18Z chrispb

### Classes

- interface [CMISS\\_PARMETIS::interface](#)

### Namespaces

- namespace [CMISS\\_PARMETIS](#)

*This module is a CMISS buffer module to the ParMETIS library.*

### Functions

- subroutine [CMISS\\_PARMETIS::PARMETIS\\_PARTKWAY](#) (VERTEX\_DISTANCE, XADJ, ADJNCY, VERTEX\_WEIGHT, ADJ\_WEIGHT, WEIGHT\_FLA, NUM\_FLAG, NCON,&NUMBER\_PARTS, TP\_WEIGHTS, UB\_VEC, OPTIONS, NUMBER\_EDGES\_CUT, PARTITION, COMMUNICATOR, ERR, ERROR,\*)

*Buffer routine to the ParMetis ParMETIS\_V3\_PartKway routine.*

- subroutine [CMISS\\_PARMETIS::PARMETIS\\_PARTMESHKWAY](#) (ELEMENT\_DISTANCE, ELEMENT\_PTR, ELEMENT\_INDEX, ELEMENT\_WEIGHT, WEIGHT\_FLAG, NUM\_FLAG, CON,&NUMBER\_COMMON\_NODES, NUMBER\_PARTS, TP\_WEIGHTS, UB\_VEC, OPTIONS, NUMBER\_EDGES\_CUT, PARTITION, COMMUNICATOR, ERR, ERROR,\*)

*Buffer routine to the ParMetis ParMETIS\_V3\_PartMeshKway routine.*

### 8.12.1 Detailed Description

#### Id

[cmiss\\_parmetis.f90](#) 177 2008-10-25 13:38:18Z chrispb

#### Author:

Chris Bradley This module is a CMISS buffer module to the ParMETIS library.

### 8.12.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [cmiss\\_parmetis.f90](#).

## 8.13 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/cmiss\_- petsc.f90 File Reference

### Id

[cmiss\\_petsc.f90](#) 178 2008-10-26 13:07:57Z chrisbradley

Include dependency graph for cmiss\_petsc.f90:

### Classes

- interface [CMISS\\_PETSC::interface](#)
- interface [CMISS\\_PETSC::PETSC\\_SNESSETJACOBIAN](#)

### Namespaces

- namespace [CMISS\\_PETSC](#)

*This module is a CMISS buffer module to the PETSc library.*

### Functions

- subroutine [CMISS\\_PETSC::PETSC\\_ERRORHANDLING\\_SET\\_OFF](#) (ERR, ERROR,\*)  
*Set PETSc error handling on.*
- subroutine [CMISS\\_PETSC::PETSC\\_ERRORHANDLING\\_SET\\_ON](#) (ERR, ERROR,\*)  
*Set PETSc error handling on.*
- subroutine [CMISS\\_PETSC::PETSC\\_FINALIZE](#) (ERR, ERROR,\*)  
*Buffer routine to the PETSc PetscFinalize routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_INITIALIZE](#) (FILE, ERR, ERROR,\*)  
*Buffer routine to the PETSc PetscInitialize routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_ISFINALISE](#) (IS\_, ERR, ERROR,\*)
- subroutine [CMISS\\_PETSC::PETSC\\_ISINITIALISE](#) (IS\_, ERR, ERROR,\*)
- subroutine [CMISS\\_PETSC::PETSC\\_ISDESTROY](#) (IS\_, ERR, ERROR,\*)  
*Buffer routine to the PETSc ISDestroy routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_ISCOLORINGFINALISE](#) (ISCOLORING, ERR, ERROR,\*)
- subroutine [CMISS\\_PETSC::PETSC\\_ISCOLORINGINITIALISE](#) (ISCOLORING, ERR, ERROR,\*)
- subroutine [CMISS\\_PETSC::PETSC\\_ISCOLORINGDESTROY](#) (ISCOLORING, ERR, ERROR,\*)  
*Buffer routine to the PETSc ISColoringDestroy routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_ISLOCALTOGLOBALMAPPINGFINALISE](#) (ISLOCALTOGLOBALMAPPING, ERR, ERROR,\*)

- subroutine [CMISS\\_PETSC::PETSC\\_ISLOCALTOGLOBALMAPPINGINITIALISE](#) (ISLOCALTOGLOBALMAPPING, ERR, ERROR,\*)
- subroutine [CMISS\\_PETSC::PETSC\\_ISLOCALTOGLOBALMAPPINGAPPLY](#) (CTX, TYPE, NIN, IDXIN, NOUT, IDXOUT, ERR, ERROR,\*)

*Buffer routine to the PETSc ISLocalToGlobalMappingApply routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_ISLOCALTOGLOBALMAPPINGAPPLYIS](#) (CTX, ISIN, ISOUT, ERR, ERROR,\*)

*Buffer routine to the PETSc ISLocalToGlobalMappingApplyIS routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_ISLOCALTOGLOBALMAPPINGCREATE](#) (COMMUNICATOR, N, GLOBALNUM, CTX, ERR, ERROR,\*)

*Buffer routine to the PETSc ISLocalToGlobalMappingCreate routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_ISLOCALTOGLOBALMAPPINGDESTROY](#) (CTX, ERR, ERROR,\*)

*Buffer routine to the PETSc ISLocalToGlobalMappingDestroy routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_KSPCREATE](#) (COMMUNICATOR, KSP\_, ERR, ERROR,\*)

*Buffer routine to the PETSc KSPCreate routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_KSPDESTROY](#) (KSP\_, ERR, ERROR,\*)

*Buffer routine to the PETSc KSPDestroy routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_KSPGETCONVERGEDREASON](#) (KSP\_, REASON, ERR, ERROR,\*)

*Buffer routine to the PETSc KSPGetConvergedReason routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_KSPGETITERATIONNUMBER](#) (KSP\_, ITERATION\_NUMBER, ERR, ERROR,\*)

*Buffer routine to the PETSc KSPGetIterationNumber routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_KSPGETPC](#) (KSP\_, PC\_, ERR, ERROR,\*)

*Buffer routine to the PETSc KSPGetPC routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_KSPGETRESIDUALNORM](#) (KSP\_, RESIDUAL\_NORM, ERR, ERROR,\*)

*Buffer routine to the PETSc KSPGetResidualNorm routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_KSPFINALISE](#) (KSP\_, ERR, ERROR,\*)

- subroutine [CMISS\\_PETSC::PETSC\\_KSPINITIALISE](#) (KSP\_, ERR, ERROR,\*)

- subroutine [CMISS\\_PETSC::PETSC\\_KSPSETFROMOPTIONS](#) (KSP\_, ERR, ERROR,\*)

*Buffer routine to the PETSc KSPSetFromOptions routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_KSPSETOPERATORS](#) (KSP\_, AMAT, PMAT, FLAG, ERR, ERROR,\*)

*Buffer routine to the PETSc KSPSetOperators routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_KSPSETTOLERANCES](#) (KSP\_, RTOL, ATOL, DTOL, MAXITS, ERR, ERROR,\*)

*Buffer routine to the PETSc KSPSetTolerances routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_KSPSETTYPE](#) (KSP\_, METHOD, ERR, ERROR,\*)
 

*Buffer routine to the PETSc KSPSetType routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_KSPSETUP](#) (KSP\_, ERR, ERROR,\*)
 

*Buffer routine to the PETSc KSPSetUp routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_KSPSOLVE](#) (KSP\_, B, X, ERR, ERROR,\*)
 

*Buffer routine to the PETSc KSPSolve routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_LOGPRINTSUMMARY](#) (COMMUNICATOR, FILE, ERR, ERROR,\*)
 

*Buffer routine to the PETSc PetscLogPrintSummary routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_MATASSEMBLYBEGIN](#) (A, ASSEMBLY\_TYPE, ERR, ERROR,\*)
 

*Buffer routine to the PETSc MatAssemblyBegin routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_MATASSEMBLYEND](#) (A, ASSEMBLY\_TYPE, ERR, ERROR,\*)
 

*Buffer routine to the PETSc MatAssemblyEnd routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_MATCREATE](#) (COMMUNICATOR, A, ERR, ERROR,\*)
 

*Buffer routine to the PETSc MatCreate routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_MATCREATEMPIAIJ](#) (COMMUNICATOR, LOCAL\_M, LOCAL\_N, GLOB\_L\_M, GLOBAL\_N, DIAG\_NUMBER\_NZ\_PERROW, DIAG\_NUMBER\_NZ\_EACHROW, &OFFDIAG\_NUMBER\_NZ\_PERROW, OFFDIAG\_NUMBER\_NZ\_EACHROW, A, ERR, ERROR,\*)
 

*Buffer routine to the PETSc MatCreateMPIAIJ routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_MATCREATEMPIDENSE](#) (COMMUNICATOR, LOCAL\_M, LOCAL\_N, GLOBAL\_M, GLOBAL\_N, MATRIX\_DATA, A, ERR, ERROR,\*)
 

*Buffer routine to the PETSc MatCreateMPIDense routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_MATCREATESEQAIJ](#) (COMMUNICATOR, M, N, NUMBER\_NZ\_PERROW, NUMBER\_NZ\_EACHROW, A, ERR, ERROR,\*)
 

*Buffer routine to the PETSc MatCreateSeqAIJ routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_MATCREATESEQDENSE](#) (COMMUNICATOR, M, N, MATRIX\_DATA, A, ERR, ERROR,\*)
 

*Buffer routine to the PETSc MatCreateSeqDense routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_MATDESTROY](#) (A, ERR, ERROR,\*)
 

*Buffer routine to the PETSc MatDestroy routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_MATFDCOLORINGCREATE](#) (A, ISCOLORING, FDCOLORING, ERR, ERROR,\*)
 

*Buffer routine to the PETSc MatFDColoringCreate routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_MATFDCOLORINGDESTROY](#) (MATFDCOLORING, ERR, ERROR,\*)

*Buffer routine to the PETSc MatFDColoringDestroy routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_MATFDCOLORINGFINALISE](#) (MATFDCOLORING, ERR, ERROR,\*)

- subroutine [CMISS\\_PETSC::PETSC\\_MATFDCOLORINGINITIALISE](#) (MATFDCOLORING, ERR, ERROR,\*)

- subroutine [CMISS\\_PETSC::PETSC\\_MATFDCOLORINGSETFROMOPTIONS](#) (MATFDCOLORING, ERR, ERROR,\*)

*Buffer routine to the PETSc MatFDColoringSetFromOptions routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_MATGETARRAY](#) (A, ARRAY, ERR, ERROR,\*)

*Buffer routine to the PETSc MatGetArray routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_MATGETCOLORING](#) (A, COLORING\_TYPE, ISCOLORING, ERR, ERROR,\*)

*Buffer routine to the PETSc MatGetColoring routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_MATGETOWNERSHIPRANGE](#) (A, FIRST\_ROW, LAST\_ROW, ERR, ERROR,\*)

*Buffer routine to the PETSc MatGetOwnershipRange routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_MATGETVALUES](#) (A, M, M\_INDICES, N, N\_INDICES, VALUES, ERR, ERROR,\*)

*Buffer routine to the PETSc MatGetValues routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_MATFINALISE](#) (MAT\_, ERR, ERROR,\*)

- subroutine [CMISS\\_PETSC::PETSC\\_MATINITIALISE](#) (MAT\_, ERR, ERROR,\*)

- subroutine [CMISS\\_PETSC::PETSC\\_MATRESTOREARRAY](#) (A, ERR, ERROR,\*)

*Buffer routine to the PETSc MatRestoreArray routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_MATSETLOCALTOGLOBALMAPPING](#) (A, CTX, ERR, ERROR,\*)

*Buffer routine to the PETSc MatSetLocalToGlobalMapping routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_MATSETOPTION](#) (A, OPTION, ERR, ERROR,\*)

*Buffer routine to the PETSc MatSetOption routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_MATSETSIZES](#) (A, LOCAL\_M, LOCAL\_N, GLOBAL\_M, GLOBAL\_N, ERR, ERROR,\*)

*Buffer routine to the PETSc MatSetSizes routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_MATSETVALUE](#) (A, ROW, COL, VALUE, INSERT\_MODE, ERR, ERROR,\*)

*Buffer routine to the PETSc MatSetValue routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_MATSETVALUES](#) (A, M, M\_INDICES, N, N\_INDICES, VALUES, INSERT\_MODE, ERR, ERROR,\*)

*Buffer routine to the PETSc MatSetValues routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_MATSETVALUELOCAL](#) (A, ROW, COL, VALUE, INSERT\_MODE, ERR, ERROR,\*)
 

*Buffer routine to the PETSc MatSetValueLocal routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_MATSETVALUESLOCAL](#) (A, M, M\_INDICES, N, N\_INDICES, VALUES, INSERT\_MODE, ERR, ERROR,\*)
 

*Buffer routine to the PETSc MatSetValuesLocal routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_MATVIEW](#) (A, V, ERR, ERROR,\*)
 

*Buffer routine to the PETSc MatView routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_MATZEROENTRIES](#) (A, ERR, ERROR,\*)
 

*Buffer routine to the PETSc MatZeroEntries routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_PCFINALISE](#) (PC\_, ERR, ERROR,\*)
 • subroutine [CMISS\\_PETSC::PETSC\\_PCINITIALISE](#) (PC\_, ERR, ERROR,\*)
 • subroutine [CMISS\\_PETSC::PETSC\\_PCSETTYPE](#) (PC\_, METHOD, ERR, ERROR,\*)
 

*Buffer routine to the PETSc PCSetType routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_SNESFINALISE](#) (SNES\_, ERR, ERROR,\*)
 • subroutine [CMISS\\_PETSC::PETSC\\_SNESINITIALISE](#) (SNES\_, ERR, ERROR,\*)
 • subroutine [CMISS\\_PETSC::PETSC\\_SNESCREATE](#) (COMMUNICATOR, SNES\_, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESCreate routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_SNESDESTROY](#) (SNES\_, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESDestroy routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_SNESGETCONVERGEDREASON](#) (SNES\_, REASON, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESGetConvergedReason routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_SNESGETFUNCTIONNORM](#) (SNES\_, FUNCTION\_NORM, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESGetFunctionNorm routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_SNESGETITERATIONNUMBER](#) (SNES\_, ITERATION\_NUMBER, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESGetIterationNumber routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_SNESLINESEARCHSET](#) (SNES\_, LINESEARCH\_TYPE, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESLineSearchSet routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_SNESLINESEARCHSETPARAMS](#) (SNES\_, ALPHA, MAXSTEP, STEPTOL, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESLineSearchSetParams routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_SNESSETFROMOPTIONS](#) (SNES\_, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESSetFromOptions routine.*

- subroutine **CMISS\_PETSC::PETSC\_SNESSETFUNCTION** (SNES\_, F, FFUNCTION, CTX, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESSetFunction routine.*
- subroutine **CMISS\_PETSC::PETSC\_SNESSETJACOBIAN\_MATFDCOLORING** (SNES\_, A, B, JFUNCTION, CTX, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESSetJacobian routine for MatFDColoring contexts.*
- subroutine **CMISS\_PETSC::PETSC\_SNESSETJACOBIAN\_SOLVER** (SNES\_, A, B, JFUNCTION, CTX, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESSetJacobian routine for solver contexts.*
- subroutine **CMISS\_PETSC::PETSC\_SNESSETTOLERANCES** (SNES\_, ABSTOL, RTOL, STOL, MAXIT, MAXF, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESSetTolerances routine.*
- subroutine **CMISS\_PETSC::PETSC\_SNESSETTRUSTREGIONTOLERANCE** (SNES\_, TRTOL, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESSetTrustRegionTolerance routine.*
- subroutine **CMISS\_PETSC::PETSC\_SNESSETTYPE** (SNES\_, METHOD, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESSetType routine.*
- subroutine **CMISS\_PETSC::PETSC\_SNESSOLVE** (SNES\_, B, X, ERR, ERROR,\*)
 

*Buffer routine to the PETSc SNESSolve routine.*
- subroutine **CMISS\_PETSC::PETSC\_VECFINALISE** (VEC\_, ERR, ERROR,\*)
 • subroutine **CMISS\_PETSC::PETSC\_VECINITIALISE** (VEC\_, ERR, ERROR,\*)
 • subroutine **CMISS\_PETSC::PETSC\_VECASSEMBLYBEGIN** (X, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecAssemblyBegin routine.*
- subroutine **CMISS\_PETSC::PETSC\_VECASSEMBLYEND** (X, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecAssemblyEnd routine.*
- subroutine **CMISS\_PETSC::PETSC\_VECCREATE** (COMMUNICATOR, X, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecCreate routine.*
- subroutine **CMISS\_PETSC::PETSC\_VECCREATEGHOST** (COMMUNICATOR, LOCAL\_SIZE, GLOBAL\_SIZE, NUMBER\_GHOST, GHOSTS, X, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecCreateGhost routine.*
- subroutine **CMISS\_PETSC::PETSC\_VECCREATEGHOSTWITHARRAY** (COMMUNICATOR, LOCAL\_SIZE, GLOBAL\_SIZE, NUMBER\_GHOST, GHOSTS, ARRAY, X, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecCreateGhostWithArray routine.*
- subroutine **CMISS\_PETSC::PETSC\_VECCREATEMPI** (COMMUNICATOR, LOCAL\_SIZE, GLOBAL\_SIZE, X, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecCreateMPI routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_VECCREATEMPIWITHARRAY](#) (COMMUNICATOR, LOCAL\_SIZE, GLOBAL\_SIZE, ARRAY, X, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecCreateMPIWithArray routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_VECCREATESEQ](#) (COMMUNICATOR, SIZE, X, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecCreateSeq routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_VECCREATESEQWITHARRAY](#) (COMMUNICATOR, SIZE, ARRAY, X, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecCreateSeqWithArray routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_VECDESTROY](#) (X, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecDestroy routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_VECDUPLICATE](#) (OLD, NEW, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecDuplicate routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_VECGETARRAY](#) (X, ARRAY, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecGetArray routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_VECGETARRAYF90](#) (X, ARRAY, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecGetArrayF90 routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_VECGETLOCALSIZE](#) (X, SIZE, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecGetLocalSize routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_VECGETOWNERSHIPRANGE](#) (X, LOW, HIGH, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecGetOwnershipRange routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_VECGETSIZE](#) (X, SIZE, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecGetSize routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_VECGETVALUES](#) (X, N, INDICES, VALUES, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecGetValues routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_VECGHOSTGETLOCALFORM](#) (G, L, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecGhostGetLocalForm routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_VECGHOSTRESTORELOCALFORM](#) (G, L, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecGhostRestoreLocalForm routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_VECGHOSTUPDATEBEGIN](#) (X, INSERT\_MODE, SCATTER\_MODE, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecGhostUpdateBegin routine.*
- subroutine [CMISS\\_PETSC::PETSC\\_VECGHOSTUPDATEEND](#) (X, INSERT\_MODE, SCATTER\_MODE, ERR, ERROR,\*)
 

*Buffer routine to the PETSc VecGhostUpdateEnd routine.*

*Buffer routine to the PETSc VecGhostUpdateEnd routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_VECRESTOREARRAY](#) (X, ERR, ERROR,\*)

*Buffer routine to the PETSc VecRestoreArray routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_VECRESTOREARRAYF90](#) (X, ARRAY, ERR, ERROR,\*)

*Buffer routine to the PETSc VecRestoreArrayF90 routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_VECSET](#) (X, VALUE, ERR, ERROR,\*)

*Buffer routine to the PETSc VecSet routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_VECSETFROMOPTIONS](#) (X, ERR, ERROR,\*)

*Buffer routine to the PETSc VecSetFromOptions routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_VECSETLOCALTOGLOBALMAPPING](#) (X, CTX, ERR, ERROR,\*)

*Buffer routine to the PETSc VecSetLocalToGlobalMapping routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_VECSETVALUES](#) (X, N, INDICES, VALUES, INSERT\_MODE, ERR, ERROR,\*)

*Buffer routine to the PETSc VecSetValues routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_VECSETVALUESLOCAL](#) (X, N, INDICES, VALUES, INSERT\_MODE, ERR, ERROR,\*)

*Buffer routine to the PETSc VecSetValuesLocal routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_VECSETSIZES](#) (X, LOCAL\_SIZE, GLOBAL\_SIZE, ERR, ERROR,\*)

*Buffer routine to the PETSc VecSetSizes routine.*

- subroutine [CMISS\\_PETSC::PETSC\\_VECVIEW](#) (X, V, ERR, ERROR,\*)

*Buffer routine to the PETSc VecView routine.*

## Variables

- INTEGER(INTG), parameter [CMISS\\_PETSC::PETSC\\_SNES\\_LINESEARCH\\_NONORMS](#) = 1
- INTEGER(INTG), parameter [CMISS\\_PETSC::PETSC\\_SNES\\_LINESEARCH\\_NO](#) = 2
- INTEGER(INTG), parameter [CMISS\\_PETSC::PETSC\\_SNES\\_LINESEARCH\\_QUADRATIC](#) = 3
- INTEGER(INTG), parameter [CMISS\\_PETSC::PETSC\\_SNES\\_LINESEARCH\\_CUBIC](#) = 4
- LOGICAL, save [CMISS\\_PETSC::PETSC\\_HANDLE\\_ERROR](#)

### 8.13.1 Detailed Description

#### Id

[cmiss\\_petsc.f90](#) 178 2008-10-26 13:07:57Z chrisbradley

#### Author:

Chris Bradley This module is a [CMISS](#) buffer module to the PETSc library.

### 8.13.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [cmiss\\_petsc.f90](#).

## **8.14 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/cmiss\_petsc\_types.f90 File Reference**

### **Id**

[cmiss\\_petsc\\_types.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

Include dependency graph for cmiss\_petsc\_types.f90:

### **Classes**

- struct [CMISS\\_PETSC\\_TYPES::PETSC\\_IS\\_TYPE](#)
- struct [CMISS\\_PETSC\\_TYPES::PETSC\\_ISLOCALTOGLOBALMAPPING\\_TYPE](#)
- struct [CMISS\\_PETSC\\_TYPES::PETSC\\_ISCOLORING\\_TYPE](#)
- struct [CMISS\\_PETSC\\_TYPES::PETSC\\_KSP\\_TYPE](#)
- struct [CMISS\\_PETSC\\_TYPES::PETSC\\_MAT\\_TYPE](#)
- struct [CMISS\\_PETSC\\_TYPES::PETSC\\_MATFDCOLORING\\_TYPE](#)
- struct [CMISS\\_PETSC\\_TYPES::PETSC\\_PC\\_TYPE](#)
- struct [CMISS\\_PETSC\\_TYPES::PETSC\\_SNES\\_TYPE](#)
- struct [CMISS\\_PETSC\\_TYPES::PETSC\\_VEC\\_TYPE](#)

### **Namespaces**

- namespace [CMISS\\_PETSC\\_TYPES](#)

*This module contains types related to the PETSc library.*

#### **8.14.1 Detailed Description**

### **Id**

[cmiss\\_petsc\\_types.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

### **Author:**

Chris Bradley This module contains type definitions related to the PETSc library.

#### **8.14.2 LICENSE**

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University

of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [cmiss\\_petsc\\_types.f90](#).

## 8.15 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/computational\_environment.f90 File Reference

### Id

[computational\\_environment.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

### Classes

- struct [COMP\\_ENVIRONMENT::CACHE\\_TYPE](#)  
*Contains information on a cache heirarchy.*
- struct [COMP\\_ENVIRONMENT::COMPUTATIONAL\\_NODE\\_TYPE](#)  
*Contains information on a computational node containing a number of processors.*
- struct [COMP\\_ENVIRONMENT::MPI\\_COMPUTATIONAL\\_NODE\\_TYPE](#)  
*Contains information on the MPI type to transfer information about a computational node.*
- struct [COMP\\_ENVIRONMENT::COMPUTATIONAL\\_ENVIRONMENT\\_TYPE](#)  
*Contains information on the computational environment the program is running in.*

### Namespaces

- namespace [COMP\\_ENVIRONMENT](#)  
*This module contains all computational environment variables.*

### Functions

- subroutine [COMP\\_ENVIRONMENT::COMPUTATIONAL\\_NODE\\_FINALISE](#)  
([COMPUTATIONAL\\_NODE](#), ERR, ERROR,\*)  
*Finalises the computational node data structures and deallocates all memory.*
- subroutine [COMP\\_ENVIRONMENT::COMPUTATIONAL\\_NODE\\_INITIALISE](#)  
([COMPUTATIONAL\\_NODE](#), RANK, ERR, ERROR,\*)  
*Initialises the computational node data structures.*
- subroutine [COMP\\_ENVIRONMENT::COMPUTATIONAL\\_NODE\\_MPI\\_TYPE\\_FINALISE](#)  
(ERR, ERROR,\*)  
*Finalises the data structure containing the MPI type information for the [COMPUTATIONAL\\_NODE\\_TYPE](#).*
- subroutine [COMP\\_ENVIRONMENT::COMPUTATIONAL\\_NODE\\_MPI\\_TYPE\\_INITIALISE](#)  
([COMPUTATIONAL\\_NODE](#), ERR, ERROR,\*)  
*Initialises the data structure containing the MPI type information for the [COMPUTATIONAL\\_NODE\\_TYPE](#).*
- subroutine [COMP\\_ENVIRONMENT::COMPUTATIONAL\\_ENVIRONMENT\\_FINALISE](#) (ERR,  
ERROR,\*)

*Finalises the computational environment data structures and deallocates all memory.*

- subroutine      [COMP\\_ENVIRONMENT::COMPUTATIONAL\\_ENVIRONMENT\\_INITIALISE](#)  
(ERR, ERROR,\*)

*Initialises the computational environment data structures.*

- INTEGER(INTG)      [COMP\\_ENVIRONMENT::COMPUTATIONAL\\_NODE\\_NUMBER\\_GET](#)  
(ERR, ERROR)

*Returns the number/rank of the computational nodes.*

- INTEGER(INTG)      [COMP\\_ENVIRONMENT::COMPUTATIONAL\\_NODES\\_NUMBER\\_GET](#)  
(ERR, ERROR)

*Returns the number of computational nodes.*

## Variables

- TYPE(COMPUTATIONAL\_ENVIRONMENT\_TYPE) [COMP\\_ENVIRONMENT::COMPUTATIONAL\\_ENVIRONMENT](#)

*The computational environment the program is running in.*

- TYPE(MPI\_COMMUNICATOR)      [COMP\\_ENVIRONMENT::MPI\\_COMMUNICATOR](#)  
[COMP\\_ENVIRONMENT::MPI\\_COMMUNICAL\\_NODE\\_TYPE\\_DATA](#)

*The MPI data on the computational nodes.*

### 8.15.1 Detailed Description

#### Id

[computational\\_environment.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

#### Author:

Chris Bradley This module contains all computational environment variables.

### 8.15.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [computational\\_environment.f90](#).

## 8.16 **d:/Users/tyu011/workspace/OpenCMISS-trunk/src/constants.f90**

### File Reference

## **8.17 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/coordinate\_routines.f90 File Reference**

### **Id**

[coordinate\\_routines.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

### **Classes**

- struct [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_PTR\\_TYPE](#)
- struct [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEMS\\_TYPE](#)
- interface [COORDINATE\\_ROUTINES::COORDINATE\\_CONVERT\\_FROM\\_RC](#)
- interface [COORDINATE\\_ROUTINES::COORDINATE\\_CONVERT\\_TO\\_RC](#)
- interface [COORDINATE\\_ROUTINES::COORDINATE\\_DELTA\\_CALCULATE](#)
- interface [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_DIMENSION\\_SET](#)
- interface [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_FOCUS\\_SET](#)
- interface [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_RADIAL\\_INTERPOLATION\\_TYPE\\_SET](#)
- interface [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_TYPE\\_SET](#)
- interface [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_ORIGIN\\_SET](#)
- interface [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_ORIENTATION\\_SET](#)
- interface [COORDINATE\\_ROUTINES::DXZ](#)
- interface [COORDINATE\\_ROUTINES::D2ZX](#)
- interface [COORDINATE\\_ROUTINES::DZX](#)
- interface [COORDINATE\\_ROUTINES::COORDINATE\\_DERIVATIVE\\_CONVERT\\_TO\\_RC](#)
- interface [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_DESTROY](#)

### **Namespaces**

- namespace [COORDINATE\\_ROUTINES](#)

*This module contains all coordinate transformation and support routines.*

### **Functions**

- REAL(DP) [COORDINATE\\_ROUTINES::COORDINATE\\_CONVERT\\_FROM\\_RC\\_DP](#)  
(COORDINATE\_SYSTEM, Z, ERR, ERROR)
- REAL(SP) [COORDINATE\\_ROUTINES::COORDINATE\\_CONVERT\\_FROM\\_RC\\_SP](#)  
(COORDINATE\_SYSTEM, Z, ERR, ERROR)
- REAL(DP) [COORDINATE\\_ROUTINES::COORDINATE\\_CONVERT\\_TO\\_RC\\_DP](#)  
(COORDINATE\_SYSTEM, X, ERR, ERROR)
- REAL(SP) [COORDINATE\\_ROUTINES::COORDINATE\\_CONVERT\\_TO\\_RC\\_SP](#)  
(COORDINATE\_SYSTEM, X, ERR, ERROR)
- REAL(DP) [COORDINATE\\_ROUTINES::COORDINATE\\_DELTA\\_CALCULATE\\_DP](#)  
(COORDINATE\_SYSTEM, X, Y, ERR, ERROR)
- subroutine [COORDINATE\\_ROUTINES::COORDINATE\\_METRICS\\_CALCULATE](#)  
(COORDINATE\_SYSTEM, JACOBIAN\_TYPE, METRICS, ERR, ERROR,\*)
- subroutine [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_NORMAL\\_CALCULATE](#)  
(COORDINATE\_SYSTEM, REVERSE, X, N, ERR, ERROR,\*)

- INTEGER(INTG) [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_DIMENSION\\_GET](#)(COORDINATE\_SYSTEM)
- REAL(DP) [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_FOCUS\\_GET](#)(COORDINATE\_SYSTEM, ERR, ERROR)
- INTEGER(INTG) [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_RADIAL\\_-INTERPOLATION\\_TYPE\\_GET](#)(COORDINATE\_SYSTEM)
- INTEGER(INTG) [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_TYPE\\_GET](#)(COORDINATE\_SYSTEM)
- subroutine [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_DIMENSION\\_SET\\_-NUMBER](#)(USER\_NUMBER, DIMENSION, ERR, ERROR,\*)
- subroutine [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_DIMENSION\\_SET\\_PTR](#)(COORDINATE\_SYSTEM, DIMENSION, ERR, ERROR,\*)
- subroutine [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_FOCUS\\_SET\\_NUMBER](#)(USER\_NUMBER, FOCUS, ERR, ERROR,\*)
- subroutine [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_FOCUS\\_SET\\_PTR](#)(COORDINATE\_SYSTEM, FOCUS, ERR, ERROR,\*)
- subroutine [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_RADIAL\\_-INTERPOLATION\\_TYPE\\_SET\\_NUMBER](#)(USER\_NUMBER, RADIAL\_INTERPOLATION\_-TYPE, ERR, ERROR,\*)
- subroutine [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_RADIAL\\_-INTERPOLATION\\_TYPE\\_SET\\_PTR](#)(COORDINATE\_SYSTEM, RADIAL\_-INTERPOLATION\_TYPE, ERR, ERROR,\*)
- subroutine [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_TYPE\\_SET\\_NUMBER](#)(USER\_NUMBER, TYPE, ERR, ERROR,\*)
- subroutine [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_TYPE\\_SET\\_PTR](#)(COORDINATE\_SYSTEM, TYPE, ERR, ERROR,\*)
- subroutine [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_ORIGIN\\_SET\\_NUMBER](#)(USER\_NUMBER, ORIGIN, ERR, ERROR,\*)
- subroutine [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_ORIGIN\\_SET\\_PTR](#)(COORDINATE\_SYSTEM, ORIGIN, ERR, ERROR,\*)
- subroutine [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_ORIENTATION\\_SET\\_-NUMBER](#)(USER\_NUMBER, ORIENTATION, ERR, ERROR,\*)
- subroutine [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_ORIENTATION\\_SET\\_PTR](#)(COORDINATE\_SYSTEM, ORIENTATION, ERR, ERROR,\*)
- subroutine [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_CREATE\\_START](#)(USER\_-NUMBER, COORDINATE\_SYSTEM, ERR, ERROR,\*)
- subroutine [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_CREATE\\_FINISH](#)(COORDINATE\_SYSTEM, ERR, ERROR,\*)
- subroutine [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_DESTROY\\_NUMBER](#)(USER\_NUMBER, ERR, ERROR,\*)
- subroutine [COORDINATE\\_ROUTINES::COORDINATE\\_SYSTEM\\_DESTROY\\_PTR](#)(COORDINATE\_SYSTEM, ERR, ERROR,\*)
- REAL(DP) [COORDINATE\\_ROUTINES::DXZ\\_DP](#)(COORDINATE\_SYSTEM, I, X, ERR, ER-ROR)
- REAL(DP) [COORDINATE\\_ROUTINES::D2ZX\\_DP](#)(COORDINATE\_SYSTEM, I, J, X, ERR, ER-ROR)
- REAL(DP) [COORDINATE\\_ROUTINES::DZX\\_DP](#)(COORDINATE\_SYSTEM, I, X, ERR, ER-ROR)
- subroutine [COORDINATE\\_ROUTINES::CO](#)(COORDINATE\_SYSTEM, PART\_DERIV\_TYPE, X, Z,&ERR, ERROR,\*)
- subroutine [COORDINATE\\_ROUTINES::CO](#)(COORDINATE\_SYSTEM, PART\_DERIV\_TYPE, X, Z,&ERR, ERROR,\*)

- subroutine **COORDINATE\_ROUTINES::COORDINATE\_DERIVATIVE\_NORM** (COORDINATE\_SYSTEM, PART\_DERIV\_INDEX, INTERPOLATED\_POINT, DERIV\_NORM, ERR, ERROR,\*)
- subroutine **COORDINATE\_ROUTINES::COORDINATE\_INTERPOLATION\_ADJUST** (COORDINATE\_SYSTEM, PARTIAL\_DERIVATIVE\_INDEX, VALUE, ERR, ERROR,\*)
- subroutine **COORDINATE\_ROUTINES::COORDINATE\_INTERPOLATION\_PARAMETERS\_ADJUST** (COORDINATE\_SYSTEM, INTERPOLATION\_PARAMETERS, ERR, ERROR,\*)
- subroutine **COORDINATE\_ROUTINES::COORDINATE\_SYSTEM\_USER\_NUMBER\_FIND** (USER\_NUMBER, COORDINATE\_SYSTEM, ERR, ERROR,\*)
- subroutine **COORDINATE\_ROUTINES::COORDINATE\_SYSTEMS\_FINALISE** (ERR, ERROR,\*)
- subroutine **COORDINATE\_ROUTINES::COORDINATE\_SYSTEMS\_INITIALISE** (ERR, ERROR,\*)

## Variables

- INTEGER(INTG), parameter **COORDINATE\_ROUTINES::COORDINATE\_RECTANGULAR\_CARTESIAN\_TYPE = 1**  
*Rectangular Cartesian coordinate system type.*
- INTEGER(INTG), parameter **COORDINATE\_ROUTINES::COORDINATE\_CYCLINDRICAL\_POLAR\_TYPE = 2**  
*Cylindrical polar coordinate system type.*
- INTEGER(INTG), parameter **COORDINATE\_ROUTINES::COORDINATE\_SPHERICAL\_POLAR\_TYPE = 3**  
*Spherical polar coordinate system type.*
- INTEGER(INTG), parameter **COORDINATE\_ROUTINES::COORDINATE\_PROLATE\_SPHEROIDAL\_TYPE = 4**  
*Prolate spheroidal coordinate system type.*
- INTEGER(INTG), parameter **COORDINATE\_ROUTINES::COORDINATE\_OBLATE\_SPHEROIDAL\_TYPE = 5**  
*Oblate spheroidal coordinate system type.*
- INTEGER(INTG), parameter **COORDINATE\_ROUTINES::COORDINATE\_NO\_RADIAL\_INTERPOLATION\_TYPE = 0**  
*No radial interpolation.*
- INTEGER(INTG), parameter **COORDINATE\_ROUTINES::COORDINATE\_RADIAL\_INTERPOLATION\_TYPE = 1**  
*r radial interpolation*
- INTEGER(INTG), parameter **COORDINATE\_ROUTINES::COORDINATE\_RADIAL\_SQUARED\_INTERPOLATION\_TYPE = 2**  
 *$r^2$  radial interpolation*
- INTEGER(INTG), parameter **COORDINATE\_ROUTINES::COORDINATE\_RADIAL\_CUBED\_INTERPOLATION\_TYPE = 3**  
 *$r^3$  radial interpolation*

- INTEGER(INTG), parameter **COORDINATE\_ROUTINES::COORDINATE\_JACOBIAN\_LINE\_TYPE** = 1  
*Line type Jacobian.*
- INTEGER(INTG), parameter **COORDINATE\_ROUTINES::COORDINATE\_JACOBIAN\_AREA\_TYPE** = 2  
*Area type Jacobian.*
- INTEGER(INTG), parameter **COORDINATE\_ROUTINES::COORDINATE\_JACOBIAN\_VOLUME\_TYPE** = 3  
*Volume type Jacobian.*
- CHARACTER(LEN=21) **COORDINATE\_ROUTINES::COORDINATE\_SYSTEMS\_TYPE\_STRING** = & (/ , & , & , & , & /)
- TYPE(COORDINATE\_SYSTEMS\_TYPE) **COORDINATE\_ROUTINES::COORDINATE\_SYSTEMS**
- TYPE(COORDINATE\_SYSTEM\_TYPE), pointer **COORDINATE\_ROUTINES::GLOBAL\_COORDINATE\_SYSTEM**

### 8.17.1 Detailed Description

#### **Id**

`coordinate_routines.f90` 177 2008-10-25 13:38:18Z chrisbradley

#### **Author:**

Chris Bradley This module contains all coordinate transformation and support routines.

### 8.17.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

#### **Contributor(s):**

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and

not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [coordinate\\_routines.f90](#).

## 8.18 **d:/Users/tyu011/workspace/OpenCMISS-trunk/src/distributed\_- matrix\_vector.f90 File Reference**

## **8.19 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/domain\_mappings.f90 File Reference**

### **Id**

[domain\\_mappings.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

### **Namespaces**

- namespace [DOMAIN\\_MAPPINGS](#)

*This module handles all domain mappings routines.*

### **Functions**

- subroutine [DOMAIN\\_MAPPINGS::DOMAIN\\_MAPPINGS\\_ADJACENT\\_DOMAIN\\_FINALISE](#) (ADJACENT\_DOMAIN, ERR, ERROR,\*)  
*Finalises the adjacent domain and deallocates all memory for a domain mapping.*
- subroutine [DOMAIN\\_MAPPINGS::DOMAIN\\_MAPPINGS\\_ADJACENT\\_DOMAIN\\_INITIALISE](#) (ADJACENT\_DOMAIN, ERR, ERROR,\*)  
*Initialise the adjacent domain for a domain mapping.*
- subroutine [DOMAIN\\_MAPPINGS::DOMAIN\\_MAPPINGS\\_LOCAL\\_FROM\\_GLOBAL\\_CALCULATE](#) (DOMAIN\_MAPPING, ERR, ERROR,\*)  
*Calculates the domain mappings local map from a domain mappings global map.*
- subroutine [DOMAIN\\_MAPPINGS::DOMAIN\\_MAPPINGS\\_MAPPING\\_FINALISE](#) (DOMAIN\_MAPPING, ERR, ERROR,\*)  
*Finalises the mapping for a domain mappings mapping and deallocates all memory.*
- subroutine [DOMAIN\\_MAPPINGS::DOMAIN\\_MAPPINGS\\_MAPPING\\_GLOBAL\\_FINALISE](#) (MAPPING\_GLOBAL\_MAP, ERR, ERROR,\*)  
*Finalises the global mapping in the given domain mappings.*
- subroutine [DOMAIN\\_MAPPINGS::DOMAIN\\_MAPPINGS\\_MAPPING\\_GLOBAL\\_INITIALISE](#) (MAPPING\_GLOBAL\_MAP, ERR, ERROR,\*)  
*Finalises the global mapping in the given domain mappings.*
- subroutine [DOMAIN\\_MAPPINGS::DOMAIN\\_MAPPINGS\\_MAPPING\\_INITIALISE](#) (DOMAIN\_MAPPING, NUMBER\_OF\_DOMAINS, ERR, ERROR,\*)  
*Initialises the mapping for a domain mappings mapping.*

### **Variables**

- INTEGER(INTG), parameter [DOMAIN\\_MAPPINGS::DOMAIN\\_LOCAL\\_INTERNAL](#) = 1  
*The domain item is internal to the domain.*

- INTEGER(INTG), parameter **DOMAIN\_MAPPINGS::DOMAIN\_LOCAL\_BOUNDARY** = 2

*The domain item is on the boundary of the domain.*

- INTEGER(INTG), parameter **DOMAIN\_MAPPINGS::DOMAIN\_LOCAL\_GHOST** = 3

*The domain item is ghosted from another domain.*

### 8.19.1 Detailed Description

#### Id

[domain\\_mappings.f90](#) 177 2008-10-25 13:38:18Z chrispb Bradley

#### Author:

Chris Bradley This module handles all domain mappings routines.

### 8.19.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [domain\\_mappings.f90](#).

## 8.20 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/elasticity\_routines.f90 File Reference

### Id

[elasticity\\_routines.f90](#) 177 2008-10-25 13:38:18Z chrispb Bradley

### Namespaces

- namespace [ELASTICITY\\_ROUTINES](#)

*This module handles all elasticity class routines.*

### Functions

- subroutine [ELASTICITY\\_ROUTINES::EL](#) (EQUATIONS\_SET, EQUATIONS\_TYPE, EQUATIONS\_SUBTYPE,&ERR, ERROR,\*)  
*Sets/changes the problem type and subtype for an elasticity equation set class.*
- subroutine [ELASTICITY\\_ROUTINES::ELASTICITYFINITEELEMENTCALCULATE](#) (EQUATIONS\_SET, ELEMENT\_NUMBER, ERR, ERROR,\*)  
*Calculates the element stiffness matrices and rhs vector for the given element number for an elasticity class finite element equation set.*
- subroutine [ELASTICITY\\_ROUTINES::ELASTICITYFINITEELEMENTJACOBIANEVALUATE](#) (EQUATIONS\_SET, ELEMENT\_NUMBER, ERR, ERROR,\*)  
*Evaluates the Jacobian for the given element number for an elasticity class finite element equation set.*
- subroutine [ELASTICITY\\_ROUTINES::ELASTICITYFINITEELEMENTRESIDUALEVALUATE](#) (EQUATIONS\_SET, ELEMENT\_NUMBER, ERR, ERROR,\*)  
*Evaluates the residual and rhs vector for the given element number for an elasticity class finite element equation set.*
- subroutine [ELASTICITY\\_ROUTINES::ELASTICITYEQUATIONSSETUP](#) (EQUATIONS\_SET, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)  
*Sets up the equations set for an elasticity equations set class.*
- subroutine [ELASTICITY\\_ROUTINES::ELASTICITYPROBLEMCLASSTYPESET](#) (PROBLEM, PROBLEM\_EQUATION\_TYPE, PROBLEM\_SUBTYPE, ERR, ERROR,\*)  
*Sets/changes the problem type and subtype for an elasticity problem class.*
- subroutine [ELASTICITY\\_ROUTINES::ELASTICITYPROBLEMSSETUP](#) (PROBLEM, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)  
*Sets up the problem for an elasticity problem class.*

### 8.20.1 Detailed Description

#### Id

[elasticity\\_routines.f90](#) 177 2008-10-25 13:38:18Z chrispb Bradley

**Author:**

Chris Bradley This module handles all elasticity routines.

## 8.20.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [elasticity\\_routines.f90](#).

## 8.21 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/electromechanics\_routines.f90 File Reference

### Id

[electromechanics\\_routines.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

### Namespaces

- namespace [ELECTROMECHANICS\\_ROUTINES](#)

*This module handles all electromechanics class routines.*

### 8.21.1 Detailed Description

#### Id

[electromechanics\\_routines.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

#### Author:

Chris Bradley This module handles all electromechanics routines.

### 8.21.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [electromechanics\\_routines.f90](#).

## 8.22 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/equations\_- mapping\_routines.f90 File Reference

### Id

[equations\\_mapping\\_routines.f90](#) 181 2008-10-26 18:59:43Z chrispbradley

### Namespaces

- namespace [EQUATIONS\\_MAPPING\\_ROUTINES](#)

*This module handles all equations mapping routines.*

### Functions

- subroutine [EQUATIONS\\_MAPPING\\_ROUTINES::EQUATIONS\\_MAPPING\\_CALCULATE](#)(EQUATIONS\_MAPPING, ERR, ERROR,\*)

*Calculates the equations/dofs mapping.*

- subroutine [EQUATIONS\\_MAPPING\\_ROUTINES::EQUATIONS\\_MAPPING\\_CREATE\\_FINISH](#)(EQUATIONS\_MAPPING, ERR, ERROR,\*)

*Finishes the process of creating an equations mapping.*

- subroutine [EQUATIONS\\_MAPPING\\_ROUTINES::EQUATIONS\\_MAPPING\\_CREATE\\_START](#)(EQUATIONS, EQUATIONS\_MAPPING, ERR, ERROR,\*)

*Finishes the process of creating an equations mapping for a problem solution.*

- subroutine [EQUATIONS\\_MAPPING\\_ROUTINES::EQUATIONS\\_MAPPING\\_CREATE\\_VALUES\\_CACHE\\_FINALISE](#)(CREATE\_VALUES\_CACHE, ERR, ERROR,\*)

*Finalises an equations mapping create values cache and deallocates all memory.*

- subroutine [EQUATIONS\\_MAPPING\\_ROUTINES::EQUATIONS\\_MAPPING\\_CREATE\\_VALUES\\_CACHE\\_INITIALISE](#)(EQUATIONS\_MAPPING, ERR, ERROR,\*)

*Initialises an equations mapping create values cache.*

- subroutine [EQUATIONS\\_MAPPING\\_ROUTINES::EQUATIONS\\_MAPPING\\_DESTROY](#)(EQUATIONS\_MAPPING, ERR, ERROR,\*)

*Destroy an equations mapping.*

- subroutine [EQUATIONS\\_MAPPING\\_ROUTINES::EQUATIONS\\_MAPPING\\_EQUATIONS\\_JACOBIAN\\_TO\\_VARIABLE\\_MAP\\_FINALISE](#)(EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP, ERR, ERROR,\*)

*Finalises a variable to equations Jacobian map and deallocates all memory.*

- subroutine [EQUATIONS\\_MAPPING\\_ROUTINES::EQUATIONS\\_MAPPING\\_EQUATIONS\\_JACOBIAN\\_TO\\_VARIABLE\\_MAP\\_INITIALISE](#)(EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP, ERR, ERROR,\*)

*Initialises a variable to equations Jacobian map.*

- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_FINALISE** (EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP, ERR, ERROR,\*)
 

*Finalise an equations matrix to variable maps and deallocate all memory.*
- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_INITIALISE** (EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP, ERR, ERROR,\*)
 

*Initialise an equations matrix to variable maps.*
- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_FINALISE** (EQUATIONS\_MAPPING, ERR, ERROR,\*)
 

*Finalises the equations mapping and deallocates all memory.*
- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_INITIALISE** (EQUATIONS, ERR, ERROR,\*)
 

*Initialises the equations mapping and deallocates all memory.*
- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_LINEAR\_MAPPING\_FINALISE** (LINEAR\_MAPPING, ERR, ERROR,\*)
 

*Finalises the equations mapping linear mapping and deallocates all memory.*
- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_LINEAR\_MAPPING\_INITIALISE** (EQUATIONS\_MAPPING, ERR, ERROR,\*)
 

*Initialises the equations mapping linear mapping.*
- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_LINEAR\_MATRICES\_COEFFICIENTS\_SET** (EQUATIONS\_MAPPING, MATRIX\_COEFFICIENTS, ERR, ERROR,\*)
 

*Sets the coefficients for the linear equations matrices in an equation set.*
- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_LINEAR\_MATRICES\_NUMBER\_SET** (EQUATIONS\_MAPPING, NUMBER\_OF\_LINEAR\_EQUATIONS\_MATRICES, ERR, ERROR,\*)
 

*Sets the mapping between the dependent field variables and the linear equations matrices.*
- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_LINEAR\_MATRICES\_VARIABLE\_TYPES\_SET** (EQUATIONS\_MAPPING, MATRIX\_VARIABLE\_TYPES, ERR, ERROR,\*)
 

*Sets the mapping between the dependent field variable types and the linear equations matrices.*
- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_NONLINEAR\_MAPPING\_FINALISE** (NONLINEAR\_MAPPING, ERR, ERROR,\*)
 

*Finalises the equations mapping nonlinear mapping and deallocates all memory.*
- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_NONLINEAR\_MAPPING\_INITIALISE** (EQUATIONS\_MAPPING, ERR, ERROR,\*)
 

*Initialises the equations mapping nonlinear mapping.*
- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_RESIDUAL\_COEFFICIENT\_SET** (EQUATIONS\_MAPPING, RESIDUAL\_COEFFICIENT, ERR, ERROR,\*)

*Sets the coefficient applied to the equations set residual vector.*

- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_RESIDUAL\_VARIABLE\_TYPE\_SET** (EQUATIONS\_MAPPING, RESIDUAL\_VARIABLE\_TYPE, ERR, ERROR,\*)

*Sets the mapping between a dependent field variable and the equations set residual vector.*

- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_RHS\_COEFFICIENT\_SET** (EQUATIONS\_MAPPING, RHS\_COEFFICIENT, ERR, ERROR,\*)

*Sets the coefficient applied to the equations set RHS vector.*

- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_RHS\_MAPPING\_FINALISE** (RHS\_MAPPING, ERR, ERROR,\*)

*Finalises the equations mapping RHS mapping and deallocates all memory.*

- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_RHS\_MAPPING\_INITIALISE** (EQUATIONS\_MAPPING, ERR, ERROR,\*)

*Initialises the equations mapping RHS mapping.*

- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_RHS\_VARIABLE\_TYPE\_SET** (EQUATIONS\_MAPPING, RHS\_VARIABLE\_TYPE, ERR, ERROR,\*)

*Sets the mapping between a dependent field variable and the equations set rhs vector.*

- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_SOURCE\_COEFFICIENT\_SET** (EQUATIONS\_MAPPING, SOURCE\_COEFFICIENT, ERR, ERROR,\*)

*Sets the coefficient applied to the equations set source vector.*

- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_SOURCE\_MAPPING\_FINALISE** (SOURCE\_MAPPING, ERR, ERROR,\*)

*Finalises the equations mapping source mapping and deallocates all memory.*

- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_SOURCE\_MAPPING\_INITIALISE** (EQUATIONS\_MAPPING, ERR, ERROR,\*)

*Initialises the equations mapping source mapping.*

- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_SOURCE\_VARIABLE\_TYPE\_SET** (EQUATIONS\_MAPPING, SOURCE\_VARIABLE\_TYPE, ERR, ERROR,\*)

*Sets the mapping between a source field variable and the equations set source vector.*

- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_VARIABLE\_TO\_EQUATIONS\_COLUMN\_MAP\_FINALISE** (VARIABLE\_TO\_EQUATIONS\_COLUMN\_MAP, ERR, ERROR,\*)

*Finalise an equations mapping equations matrix map.*

- subroutine **EQUATIONS\_MAPPING\_ROUTINES::EQUATIONS\_MAPPING\_VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_MAP\_FINALISE** (VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_MAP, ERR, ERROR,\*)

*Finalises a variable to equations Jacobian map and deallocates all memory.*

- subroutine `EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_VARIABLE_TO_EQUATIONS_JACOBIAN_MAP_INITIALISE` (VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_MAP, ERR, ERROR,\*)

*Initialises a variable to equations Jacobian map.*

- subroutine `EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_VARIABLE_TO_EQUATIONS_MATRICES_MAP_FINALISE` (VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP, ERR, ERROR,\*)

*Finalises a variable to equations matrices map and deallocates all memory.*

- subroutine `EQUATIONS_MAPPING_ROUTINES::EQUATIONS_MAPPING_VARIABLE_TO_EQUATIONS_MATRICES_MAP_INITIALISE` (VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP, ERR, ERROR,\*)

*Initialise an equations mapping equations matrix map.*

### 8.22.1 Detailed Description

#### Id

[equations\\_mapping\\_routines.f90](#) 181 2008-10-26 18:59:43Z chrisbradley

#### Author:

Chris Bradley This module handles all equations mapping routines.

### 8.22.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [equations\\_mapping\\_routines.f90](#).

## 8.23 **d:/Users/tyu011/workspace/OpenCMISS-trunk/src/equations\_- matrices\_routines.f90 File Reference**

## 8.24 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/equations\_set\_constants.f90 File Reference

### Id

[equations\\_set\\_constants.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

### Namespaces

- namespace EQUATIONS\_SET\_CONSTANTS

*This module defines all constants shared across equations set routines.*

### Variables

- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_NO\_CLASS = 0
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_ELASTICITY\_CLASS = 1
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_FLUID\_MECHANICS\_CLASS = 2
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_ELECTROMAGNETICS\_CLASS = 3
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_CLASSICAL\_FIELD\_CLASS = 4
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_MODAL\_CLASS = 5
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_FITTING\_CLASS = 6
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_OPTIMISATION\_CLASS = 7
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_NO\_TYPE = 0
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_LINEAR\_ELASTICITY\_TYPE = 1
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SETFINITE\_ELASTICITY\_TYPE = 2
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_STOKES\_FLUID\_TYPE = 1
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_NAVIER\_STOKES\_FLUID\_TYPE = 2
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_ELECTROSTATIC\_TYPE = 1
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_MAGNETOSTATIC\_TYPE = 2
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_MAXWELLS\_EQUATIONS\_TYPE = 3
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_LAPLACE\_EQUATION\_TYPE = 1
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_POISSON\_EQUATION\_TYPE = 2

- INTEGER(INTG), parameter    EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
HELMHOLTZ\_EQUATION\_TYPE = 3
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_WAVE\_-  
EQUATION\_TYPE = 4
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
DIFFUSION\_EQUATION\_TYPE = 5
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
ADVECTION\_DIFFUSION\_EQUATION\_TYPE = 6
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
REACTION\_DIFFUSION\_EQUATION\_TYPE = 7
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
BIHARMONIC\_EQUATION\_TYPE = 8
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_LINEAR\_-  
ELASTIC\_MODAL\_TYPE = 1
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_NO\_-  
SUBTYPE = 0
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
STANDARD\_LAPLACE\_SUBTYPE = 1
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
GENERALISED\_LAPLACE\_SUBTYPE = 2
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
CONSTANT\_SOURCE\_POISSON\_SUBTYPE = 1
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_LINEAR\_-  
SOURCE\_POISSON\_SUBTYPE = 2
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_-  
QUADRATIC\_SOURCE\_POISSON\_SUBTYPE = 2
- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_SETUP\_-  
INITIAL\_TYPE = 1

*Initial setup.*

- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_SETUP\_-  
GEOMETRY\_TYPE = 2

*Geometry setup.*

- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_SETUP\_-  
DEPENDENT\_TYPE = 3

*Dependent variables.*

- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_SETUP\_-  
MATERIALS\_TYPE = 4

*Materials setup.*

- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_SETUP\_-  
SOURCE\_TYPE = 5

*Source setup.*

- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_SETUP\_-  
SOURCE\_MATERIALS\_TYPE = 6

*Source materials setup.*

- INTEGER(INTG), parameter EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_SETUP\_-  
ANALYTIC\_TYPE = 7

*Analytic setup.*

- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_SETUP\_FIXED\_CONDITIONS\_TYPE = 8**

*Fixed conditions.*

- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_SETUP\_EQUATIONS\_TYPE = 9**

*Equations setup.*

- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_SETUP\_FINAL\_TYPE = 9**

*Final setup.*

- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_SETUP\_START\_ACTION = 1**

*Start setup action.*

- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_SETUP\_FINISH\_ACTION = 2**

*Finish setup action.*

- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_NOT\_FIXED = 0**

*The dof is not fixed.*

- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_FIXED\_BOUNDARY\_CONDITION = 1**

*The dof is fixed as a boundary condition.*

- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_MIXED\_BOUNDARY\_CONDITION = 2**

*The dof is set as a mixed boundary condition.*

- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::NUMBER\_OF\_EQUATIONS\_SET\_LINEARITY\_TYPES = 3**

*The number of problem linearity types defined.*

- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_LINEAR = 1**

*The problem is linear.*

- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_NONLINEAR = 2**

*The problem is non-linear.*

- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_NONLINEAR\_BCS = 3**

*The problem has non-linear boundary conditions.*

- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_JACOBIAN\_NOT\_CALCULATED** = 1  
*The Jacobian values will not be calculated for the nonlinear equations set.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_JACOBIAN\_ANALYTIC\_CALCULATED** = 2  
*The Jacobian values will be calculated analytically for the nonlinear equations set.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_JACOBIAN\_FD\_CALCULATED** = 3  
*The Jacobian values will be calculated using finite differences for the nonlinear equations set.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::NUMBER\_OF\_EQUATIONS\_SET\_TIME\_TYPES** = 3  
*The number of problem time dependence types defined.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_STATIC** = 1  
*The problem is static and has no time dependence.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_DYNAMIC** = 2  
*The problem is dynamic.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_QUASISTATIC** = 3  
*The problem is quasi-static.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::NUMBER\_OF\_EQUATIONS\_SET SOLUTION\_METHODS** = 6  
*The number of solution methods defined.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_FEM SOLUTION\_METHOD** = 1  
*Finite Element Method solution method.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_BEM SOLUTION\_METHOD** = 2  
*Boundary Element Method solution method.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_FD SOLUTION\_METHOD** = 3  
*Finite Difference solution method.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_FV SOLUTION\_METHOD** = 4  
*Finite Volume solution method.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_GFEM SOLUTION\_METHOD** = 5  
*Grid-based Finite Element Method solution method.*

- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_GFV\_-SOLUTION\_METHOD = 6**

*Grid-based Finite Volume solution method.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_NO\_-OUTPUT = 0**

*No output.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_TIMING\_-OUTPUT = 1**

*Timing information output.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_MATRIX\_-OUTPUT = 2**

*All below and equation matrices output.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_ELEMENT\_MATRIX\_OUTPUT = 3**

*All below and Element matrices output.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_SPARSE\_-MATRICES = 1**

*Use sparse matrices for the equations set.*
- INTEGER(INTG), parameter **EQUATIONS\_SET\_CONSTANTS::EQUATIONS\_SET\_FULL\_-MATRICES = 2**

*Use fully populated matrices for the equations set.*

### **8.24.1 Detailed Description**

#### **Id**

[equations\\_set\\_constants.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

#### **Author:**

Chris Bradley This module handles all constants shared across equations set routines.

### **8.24.2 LICENSE**

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [equations\\_set\\_constants.f90](#).

**8.25 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/equations\_set\_routines.f90 File Reference**

## **8.25 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/equations\_set\_routines.f90 File Reference**

## 8.26 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/f90c\_c.c File Reference

Include dependency graph for f90c\_c.c:

### Typedefs

- `typedef int integer`
- `typedef integer logical`

### Functions

- `void CStringLen (int *length, char *string)`
- `void PackCharacters (integer *integer_char, integer *char_num, char *integer_string)`
- `void UnPackCharacters (integer *integer_char, integer *char_num, char *integer_string)`

#### 8.26.1 Typedef Documentation

##### 8.26.1.1 `typedef int integer`

Definition at line 98 of file f90c\_c.c.

##### 8.26.1.2 `typedef integer logical`

Definition at line 99 of file f90c\_c.c.

#### 8.26.2 Function Documentation

##### 8.26.2.1 `void CStringLen (int * length, char * string)`

Definition at line 118 of file f90c\_c.c.

##### 8.26.2.2 `void PackCharacters (integer * integer_char, integer * char_num, char * integer_string)`

Definition at line 124 of file f90c\_c.c.

##### 8.26.2.3 `void UnPackCharacters (integer * integer_char, integer * char_num, char * integer_string)`

Definition at line 140 of file f90c\_c.c.

## 8.27 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/f90c\_f.f90 File Reference

### Id

[f90c\\_f.f90](#) 177 2008-10-25 13:38:18Z chrispb Bradley

### Classes

- interface [F90C::interface](#)

### Namespaces

- namespace [F90C](#)

*This module handles calling C from Fortran90 and vice versa. It needs to be linked with the [f90c\\_c.c](#) module.*

### Functions

- INTEGER(INTG) [F90C::CSTRINGLENGTH](#) (CSTRING)
- subroutine [F90C::C2FSTRING](#) (CSTRING, FSTRING, ERR, ERROR,\*)
- INTEGER(INTG) [F90C::FSTRINGLENGTH](#) (FSTRING)
- subroutine [F90C::F2CSTRING](#) (CSTRING, FSTRING, ERR, ERROR,\*, FSTRINGLEN)

#### 8.27.1 Detailed Description

### Id

[f90c\\_f.f90](#) 177 2008-10-25 13:38:18Z chrispb Bradley

### Author:

Chris Bradley This module handles calling C from Fortran90 and vice versa. It needs to be linked with the [f90c\\_c.c](#) module.

#### 8.27.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [f90c\\_f.f90](#).

## 8.28 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/field\_IO\_routines.f90 File Reference

### Id

[field\\_IO\\_routines.f90](#) 181 2008-10-26 18:59:43Z chrisbradley

### Classes

- struct [FIELD\\_IO\\_ROUTINES::MESH\\_ELEMENTS\\_TYPE\\_PTR\\_TYPE](#)  
*field variable component type pointer for IO*
- struct [FIELD\\_IO\\_ROUTINES::FIELD\\_VARIABLE\\_COMPONENT\\_PTR\\_TYPE](#)  
*field variable component type pointer for IO*
- struct [FIELD\\_IO\\_ROUTINES::FIELD\\_IO\\_INFO\\_SET](#)  
*contains information for parallel IO, and it is nodal base*
- struct [FIELD\\_IO\\_ROUTINES::FIELD\\_IO\\_NODAL\\_INFO\\_SET](#)  
*contains information for parallel IO, and it is nodal base*
- struct [FIELD\\_IO\\_ROUTINES::FIELD\\_IO\\_ELEMENTALL\\_INFO\\_SET](#)  
*contains information for parallel IO, and it is nodal base*

### Namespaces

- namespace [FIELD\\_IO\\_ROUTINES](#)  
*Implements lists of Field IO operation.*

### Functions

- subroutine [FIELD\\_IO\\_ROUTINES::FIELD\\_IO\\_FIELD\\_INFO](#) (STRING, LABEL\_TYPE, FIELD\_TYPE, ERR, ERROR,\*)  
*Get the field information.*
- INTEGER(INTG) [FIELD\\_IO\\_ROUTINES::FIELD\\_IO\\_DERIVATIVE\\_INFO](#) (LINE, ERR, ERROR)  
*Get the derivative information.*
- subroutine [FIELD\\_IO\\_ROUTINES::FIELD\\_IO\\_CREATE\\_FIELDS](#)  
*Create decomposition.*
- subroutine [FIELD\\_IO\\_ROUTINES::FIE](#) (DECOMPOSITION, DECOMPOSITION\_USER\_NUMBER, DECOMPOSITION\_METHOD, MESH, NUMBER\_OF\_DOMAINS,&ERR, ERROR,\*)  
*Create decompositon.*

- subroutine [FIELD\\_IO\\_ROUTINES::FIELD\\_IO\\_FILEDS\\_IMPORT](#) (NAME, METHOD, REGION, MESH, MESH\_USER\_NUMBER, DECOMPOSITION, DECOMPOSITION\_USER\_NUMBER,&DECOMPOSITION\_METHOD, FIELD\_VALUES\_SET\_TYPE, FIELD\_SCALING\_TYPE, ERR, ERROR,\*)

*Import fields from files into different computational nodes.*

- subroutine [FIELD\\_IO\\_ROUTINES::FIELD\\_IO\\_FILL\\_BASIS\\_INFO](#) (INTERPOLATION\_XI, LIST\_STR, NUMBER\_OF\_COMPONENTS, ERR, ERROR,\*)

*Finding basis information.*

- subroutine [FIELD\\_IO\\_ROUTINES::FIELD\\_IO\\_IMPORT\\_GLOBAL\\_MESH](#) (NAME, REGION, MESH, MESH\_USER\_NUMBER, MA TER\_COMPUTATIONAL\_NUMBER,&my\_computational\_node\_number,&MESH\_COMPONENTS\_OF\_FIELD\_COMPONENTS,&COMPONENTS\_IN\_FIELDS, NUMBER\_OF\_FIELDS, NUMBER\_OF\_EXNODE\_FILES, ERR, ERROR,\*)

*Read the global mesh into one computational node first and then broadcasting to others nodes.*

- subroutine [FIELD\\_IO\\_ROUTINES::FIELD\\_IO\\_TRANSLATE\\_LABEL\\_INTO\\_INTERPOLATION\\_TYPE](#) (INTERPOLATION, LABEL\_TYPE, ERR, ERROR,\*)

*Finding basis information.*

- subroutine [FIELD\\_IO\\_ROUTINES::FIELD\\_IO\\_ELEMENTS\\_EXPORT](#) (FIELDS, FILE\_NAME, METHOD, ERR, ERROR,\*)

*Export elemental information into multiple files.*

- TYPE(VARYING\_STRING) [FIELD\\_IO\\_ROUTINES::FIELD\\_IO\\_BASIS\\_LHTP\\_FAMILY\\_LABEL](#) (BASIS, num\_scl, num\_node, LABEL\_TYPE, ERR, ERROR)

*Finding basis information.*

- subroutine [FIELD\\_IO\\_ROUTINES::FIELD\\_IO\\_EXPORT\\_ELEMENTAL\\_GROUP\\_HEADER\\_FORTRAN](#) (LOCAL\_PROCESS\_ELEMENTAL\_INFO\_SET, LOCAL ELEMENTAL\_NUMBER, MAX\_NODE\_CPMP\_INDEX,&NUM\_OF\_SCALING\_FACTOR\_SETS, LIST\_COMP\_SCALE, my\_computational\_node\_number, FILE\_ID, ERR, ERROR,\*)

*Write the header of a group elements using FORTRAN.*

- subroutine [FIELD\\_IO\\_ROUTINES::FIELD\\_IO\\_EXPORT\\_ELEMENTS\\_INTO\\_](#) (LOCAL\_PROCESS\_ELEMENTAL\_INFO\_SET, NAME, my\_computational\_node\_number,&computational\_node\_numbers, ERR, ERROR,\*)

*Write all the elemental information from LOCAL\_PROCESS\_NODAL\_INFO\_SET to exelem files.*

- subroutine [FIELD\\_IO\\_ROUTINES::FIELD\\_IO\\_ELEMENTAL\\_INFO\\_SET\\_SORT](#) (LOCAL\_PROCESS\_ELEMENTAL\_INFO\_SET, my\_computational\_node\_number, ERR, ERROR,\*)

*Sort the Elemental\_info\_set according to the type of field variable components.*

- subroutine [FIELD\\_IO\\_ROUTINES::FIELD\\_IO\\_ELEMENTAL\\_INFO\\_SET\\_ATTACH\\_LOCAL\\_PROCESS](#) (LOCAL\_PROCESS\_ELEMENTAL\_INFO\_SET, ERR, ERROR,\*)

*Collect the elemental information from each MPI process.*

- subroutine [FIELD\\_IO\\_ROUTINES::FIELD\\_IO\\_ELEMENTAL\\_INFO\\_SET\\_FINALIZE](#) (LOCAL\_PROCESS\_ELEMENTAL\_INFO\_SET, ERR, ERROR,\*)

*Finalized the elemental information set.*

- subroutine **FIELD\_IO\_ROUTINES::FIELD\_IO\_ELEMENTAL\_INFO\_SET\_INITIALISE** (LOCAL\_PROCESS\_ELEMENTAL\_INFO\_SET, FIELDS, ERR, ERROR,\*)  
*Initialize the elemental information set.*
- subroutine **FIELD\_IO\_ROUTINES::FIELD\_IO\_NODES\_EXPORT** (FIELDS, FILE\_NAME, METHOD, ERR, ERROR,\*)  
*Export nodal information.*
- subroutine **FIELD\_IO\_ROUTINES::FIELD\_IO\_NODAL\_INFO\_SET\_INITIALISE** (LOCAL\_PROCESS\_NODAL\_INFO\_SET, FIELDS, ERR, ERROR,\*)  
*Initialize nodal information set.*
- subroutine **FIELD\_IO\_ROUTINES::FIELD\_IO\_NODAL\_INFO\_SET\_ATTACH\_LOCAL\_PROCESS** (LOCAL\_PROCESS\_NODAL\_INFO\_SET, ERR, ERROR,\*)  
*Collect nodal information from each MPI process.*
- subroutine **FIELD\_IO\_ROUTINES::FIELD\_IO\_NODAL\_INFO\_SET\_SORT** (LOCAL\_PROCESS\_NODAL\_INFO\_SET, my\_computational\_node\_number, ERR, ERROR,\*)  
*Sort nodal information according to the type of field variable component.*
- subroutine **FIELD\_IO\_ROUTINES::FIELD\_IO\_NODAL\_INFO\_SET\_FINALIZE** (LOCAL\_PROCESS\_NODAL\_INFO\_SET, ERR, ERROR,\*)  
*Finalize nodal information set.*
- TYPE(VARYING\_STRING) **FIELD\_IO\_ROUTINES::FIELD\_IO\_LABEL\_DERIVATIVE\_INFO\_GET** (GROUP\_DERIVATIVES, NUMBER\_DERIVATIVES, LABEL\_TYPE, ERR, ERROR)  
*Get the derivative information.*
- TYPE(VARYING\_STRING) **FIELD\_IO\_ROUTINES::FIELD\_IO\_LABEL\_FIELD\_INFO\_GET** (COMPONENT, LABEL\_TYPE, ERR, ERROR)  
*Get the field information.*
- subroutine **FIELD\_IO\_ROUTINES::FIELD\_IO\_EXPORT\_NODAL\_GROUP\_HEADER\_FORTRAN** (LOCAL\_PROCESS\_NODAL\_INFO\_SET, LOCAL\_NODAL\_NUMBER, MAX\_NUM\_OF\_NODAL\_DERIVATIVES, &my\_computational\_node\_number, FILE\_ID, ERR, ERROR,\*)  
*Write the header of a group nodes using FORTRAN.*
- subroutine **FIELD\_IO\_ROUTINES::FIELD\_IO\_EXPORT\_NODES\_INTO\_LOC** (LOCAL\_PROCESS\_NODAL\_INFO\_SET, NAME, my\_computational\_node\_number, &computational\_node\_numbers, ERR, ERROR,\*)  
*Write all the nodal information from LOCAL\_PROCESS\_NODAL\_INFO\_SET to local exnode files.*
- TYPE(VARYING\_STRING) **FIELD\_IO\_ROUTINES::FIELD\_IO\_FILEDS\_GROUP\_INFO\_GET** (FIELDS, ERR, ERROR)  
*Get the region label.*
- TYPE(VARYING\_STRING) **FIELD\_IO\_ROUTINES::FIELD\_IO\_MULTI\_FILES\_INFO\_GET** (computational\_node\_numbers, ERR, ERROR)

*Get the number of files.*

- subroutine `FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_STRING` (FILE\_ID, STRING\_DATA, FILE\_END, ERR, ERROR,\*)

*Read a string using FORTRAN IO.*

- subroutine `FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_STRING` (FILE\_ID, STRING\_DATA, LEN\_OF\_DATA, ERR, ERROR,\*)

*Write a string using FORTRAN IO.*

- subroutine `FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_DP` (FILE\_ID, REAL\_DATA, LEN\_OF\_DATA, FILE\_END, ERR, ERROR,\*)

*Read a real data using FORTRAN IO.*

- subroutine `FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_DP` (FILE\_ID, REAL\_DATA, LEN\_OF\_DATA, ERR, ERROR,\*)

*Write a real data using FORTRAN IO.*

- subroutine `FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_INTG` (FILE\_ID, INTG\_DATA, LEN\_OF\_DATA, ERR, ERROR,\*)

*Read a integer data.*

- subroutine `FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_INTG` (FILE\_ID, INTG\_DATA, LEN\_OF\_DATA, ERR, ERROR,\*)

*Write a integer data.*

- subroutine `FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_OPEN` (FILE\_ID, FILE\_NAME, FILE\_STATUS, ERR, ERROR,\*)

*Open a file using Fortran.*

- subroutine `FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_REWIND` (FILE\_ID, ERR, ERROR,\*)

*Close a file using FORTRAN IO to rewind.*

- subroutine `FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_CLOSE` (FILE\_ID, ERR, ERROR,\*)

*Close a file using Fortran.*

- subroutine `FIELD_IO_ROUTINES::STRING_TO_MUTI_INTEGERS_VS` (STRING, NUMBER\_OF\_INTEGERS, INTG\_DATA, ERR, ERROR,\*)

- subroutine `FIELD_IO_ROUTINES::STRING_TO_MUTI_REALS_VS` (STRING, NUMBER\_OF\_REALS, REAL\_DATA, POSITION, ERR, ERROR,\*)

## Variables

- INTEGER(INTG), parameter `FIELD_IO_ROUTINES::SHAPE_SIZE` = 3

*size of shape*

- INTEGER(INTG), parameter `FIELD_IO_ROUTINES::FIELD_IO_FIELD_LABEL` = 1

*Type for label.*

- INTEGER(INTG), parameter `FIELD_IO_ROUTINES::FIELD_IO_VARIABLE_LABEL` = 2
- INTEGER(INTG), parameter `FIELD_IO_ROUTINES::FIELD_IO_COMPONENT_LABEL` = 3
- INTEGER(INTG), parameter `FIELD_IO_ROUTINES::FIELD_IO_DERIVATIVE_LABEL` = 4
- INTEGER(INTG), parameter `FIELD_IO_ROUTINES::FIELD_IO_SCALE_FACTORS_NUMBER_TYPE` = 5

*Type of scale factor.*

- INTEGER(INTG), parameter `FIELD_IO_ROUTINES::FIELD_IO_SCALE_FACTORS_PROPERTY_TYPE` = 6

## 8.28.1 Detailed Description

### Id

`field_IO_routines.f90` 181 2008-10-26 18:59:43Z chrispb Bradley

### Author:

Heye Zhang ThiS module handles parallel Io. Using mpi2 and parall print function, formatted text and binary IO are supported in openCMISS

## 8.28.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file `field_IO_routines.f90`.

## 8.29 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/field\_routines.f90 File Reference

### Id

[field\\_routines.f90](#) 180 2008-10-26 16:45:43Z chrisbradley

### Classes

- interface [FIELD\\_ROUTINES::FIELD\\_COMPONENT\\_INTERPOLATION\\_SET](#)
- interface [FIELD\\_ROUTINES::FIELD\\_COMPONENT\\_MESH\\_COMPONENT\\_SET](#)
- interface [FIELD\\_ROUTINES::FIELD\\_DEPENDENT\\_TYPE\\_SET](#)
- interface [FIELD\\_ROUTINES::FIELD\\_DIMENSION\\_SET](#)
- interface [FIELD\\_ROUTINES::FIELD\\_GEOMETRIC\\_FIELD\\_SET](#)
- interface [FIELD\\_ROUTINES::FIELD\\_MESH\\_DECOMPOSITION\\_SET](#)
- interface [FIELD\\_ROUTINES::FIELD\\_NUMBER\\_OF\\_COMPONENTS\\_SET](#)
- interface [FIELD\\_ROUTINES::FIELD\\_NUMBER\\_OF\\_VARIABLES\\_SET](#)
- interface [FIELD\\_ROUTINES::FIELD\\_SCALING\\_TYPE\\_SET](#)
- interface [FIELD\\_ROUTINES::FIELD\\_TYPE\\_SET](#)

### Namespaces

- namespace [FIELD\\_ROUTINES](#)

*This module handles all field related routines.*

### Functions

- INTEGER(INTG) [FIELD\\_ROUTINES::FIELD\\_COMPONENT\\_INTERPOLATION\\_GET](#) (FIELD, FIELD\_VARIABLE\_NUMBER, FIELD\_COMPONENT\_NUMBER, ERR, ERROR)  
*Gets the interpolation type for a field variable component identified by a pointer.*
- subroutine [FIELD\\_ROUTINES::FIELD\\_COMPONENT\\_INTER](#) (USER\_NUMBER, FIELD\_VARIABLE\_NUMBER, FIELD\_COMPONENT\_NUMBER, REGION,&INTERPOLATION\_TYPE, ERR, ERROR,\*)  
*Sets/changes the interpolation type for a field variable component identified by a user number and component number on a region.*
- subroutine [FIELD\\_ROUTINES::FI](#) (FIELD, FIELD\_VARIABLE\_NUMBER, FIELD\_COMPONENT\_NUMBER, INTERPOLATION\_TYPE,&ERR, ERROR,\*)  
*Sets/changes the interpolation type for a field variable component identified by a pointer.*
- INTEGER(INTG) [FIELD\\_ROUTINES::FIELD\\_COMPONENT\\_MESH\\_COMPONENT\\_GET](#) (FIELD, FIELD\_VARIABLE\_NUMBER, FIELD\_COMPONENT\_NUMBER, ERR, ERROR)  
*Gets the mesh component number for a field variable component identified by a pointer to a field and a field variable number.*
- subroutine [FIELD\\_ROUTINES::FIELD\\_COMPONENT\\_MESH\\_COM](#) (USER\_NUMBER, FIELD\_VARIABLE\_NUMBER, FIELD\_COMPONENT\_NUMBER, REGION,&MESH\_COMPONENT\_NUMBER, ERR, ERROR,\*)

*Sets/changes the mesh component number for a field variable component identified by a user number, component number and variable number on a region.*

- subroutine **FIELD\_ROUTINES::FIELD\_VARIABLE\_COMPONENT\_FINALISE** (FIELD\_VARIABLE\_COMPONENT, ERR, ERROR,\*)
 

*Finalises a field variable component and deallocates all memory.*
- subroutine **FIELD\_ROUTINES::FIELD\_VARIABLE\_COMPONENT\_INITIALISE** (FIELD, VARIABLE\_NUMBER, COMPONENT\_NUMBER, ERR, ERROR,\*)
 

*Initialises a field variable component.*
- subroutine **FIELD\_ROUTINES::FIELD\_CREATE\_FINISH** (REGION, FIELD, ERR, ERROR,\*)
 

*Finishes the creation of a field on a region.*
- subroutine **FIELD\_ROUTINES::FIELD\_CREATE\_VALUES\_CACHE\_FINALISE** (FIELD, ERR, ERROR,\*)
 

*Finalise the create values cache for a field.*
- subroutine **FIELD\_ROUTINES::FIELD\_CREATE\_VALUES\_CACHE\_INITIALISE** (FIELD, ERR, ERROR,\*)
 

*Initialises the create values cache for a field.*
- INTEGER(INTG) **FIELD\_ROUTINES::FIELD\_DEPENDENT\_TYPE\_GET** (FIELD, ERR, ERROR)
 

*Gets the dependent type for a field indentified by a pointer.*
- subroutine **FIELD\_ROUTINES::FIELD\_DEPENDENT\_TYPE\_SET\_NUMBER** (USER\_NUMBER, REGION, DEPENDENT\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the dependent type for a field identified by a user number.*
- subroutine **FIELD\_ROUTINES::FIELD\_DEPENDENT\_TYPE\_SET\_PTR** (FIELD, DEPENDENT\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the dependent type for a field indentified by a pointer.*
- subroutine **FIELD\_ROUTINES::FIELD\_DESTROY** (FIELD, ERR, ERROR,\*)
 

*Destroys a field identified by a pointer to a field.*
- INTEGER(INTG) **FIELD\_ROUTINES::FIELD\_DIMENSION\_GET** (FIELD, ERR, ERROR)
 

*Gets the field dimension for a field identified by a pointer.*
- subroutine **FIELD\_ROUTINES::FIELD\_DIMENSION\_SET\_NUMBER** (USER\_NUMBER, REGION, FIELD\_DIMENSION, ERR, ERROR,\*)
 

*Sets/changes the field dimension for a field identified by a user number.*
- subroutine **FIELD\_ROUTINES::FIELD\_DIMENSION\_SET\_PTR** (FIELD, FIELD\_DIMENSION, ERR, ERROR,\*)
 

*Sets/changes the field dimension for a field identified by a pointer.*
- subroutine **FIELD\_ROUTINES::FIELD\_INTERPOLATE\_GAUSS** (PARTIAL\_DERIVATIVE\_TYPE, QUADRATURE\_SCHEME, GAUSS\_POINT\_NUMBER, INTERPOLATED\_POINT, ERR, ERROR,\*)

*Interpolates a field at a gauss point to give an interpolated point. PARTIAL\_DERIVATIVE\_TYPE controls which partial derivatives are evaluated. If it is NO\_PART\_DERIV then only the field values are interpolated. If it is FIRST\_PART\_DERIV then the field values and first partial derivatives are interpolated. If it is SECOND\_PART\_DERIV the the field values and first and second partial derivatives are evaluated. Old CMISS name XEXG, ZEXG.*

- subroutine **FIELD\_ROUTINES::FIELD\_INTERPOLATE\_XI** (PARTIAL\_DERIVATIVE\_TYPE, XI, INTERPOLATED\_POINT, ERR, ERROR,\*)

*Interpolates a field at a xi location to give an interpolated point. XI is the element location to be interpolated at. PARTIAL\_DERIVATIVE\_TYPE controls which partial derivatives are evaluated. If it is NO\_PART\_DERIV then only the field values are interpolated. If it is FIRST\_PART\_DERIV then the field values and first partial derivatives are interpolated. If it is SECOND\_PART\_DERIV the the field values and first and second partial derivatives are evaluated. Old CMISS name PXI.*

- subroutine **FIELD\_ROUTINES::FIELD\_INTERPOLATED\_POINT\_FINALISE** (INTERPOLATED\_POINT, ERR, ERROR,\*)

*Finalises the interpolated point and deallocates all memory.*

- subroutine **FIELD\_ROUTINES::FIELD\_INTERPOLATED\_POINT\_INITIALISE** (INTERPOLATION\_PARAMETERS, INTERPOLATED\_POINT, ERR, ERROR,\*)

*Initialises the interpolated point for an interpolation parameters.*

- subroutine **FIELD\_ROUTINES::FIELD\_INTERPOLATED\_POINT\_METRICS\_CALCULATE** (JACOBIAN\_TYPE, INTERPOLATED\_POINT\_METRICS, ERR, ERROR,\*)

*Calculates the interpolated point metrics and the associated interpolated point.*

- subroutine **FIELD\_ROUTINES::FIELD\_INTERPOLATED\_POINT\_METRICS\_FINALISE** (INTERPOLATED\_POINT\_METRICS, ERR, ERROR,\*)

*Finalises the interpolated point metrics and deallocate all memory.*

- subroutine **FIELD\_ROUTINES::FIELD\_INTERPOLATED\_POINT\_METRICS\_INITIALISE** (INTERPOLATED\_POINT, INTERPOLATED\_POINT\_METRICS, ERR, ERROR,\*)

*Initialises the interpolated point metrics for an interpolated point.*

- subroutine **FIELD\_ROUTINES::FIELD\_INTERPOLATION\_PARAMETERS\_ELEMENT\_GET** (PARAMETER\_SET\_NUMBER, ELEMENT\_NUMBER, INTERPOLATION\_PARAMETERS, ERR, ERROR,\*)

*Gets the interpolation parameters for a particular element. Old CMISS name XPXE, ZPZE.*

- subroutine **FIELD\_ROUTINES::FIELD\_INTERPOLATION\_PARAMETERS\_FINALISE** (INTERPOLATION\_PARAMETERS, ERR, ERROR,\*)

*Finalises the interpolation parameters and deallocate all memory.*

- subroutine **FIELD\_ROUTINES::FIELD\_INTERPOLATION\_PARAMETERS\_INITIALISE** (FIELD, VARIABLE\_NUMBER, INTERPOLATION\_PARAMETERS, ERR, ERROR,\*)

*Initialises the interpolation parameters for a field variable.*

- subroutine **FIELD\_ROUTINES::FIELD\_INTERPOLATION\_PARAMETERS\_LINE\_GET** (PARAMETER\_SET\_NUMBER, LINE\_NUMBER, INTERPOLATION\_PARAMETERS, ERR, ERROR,\*)

*Gets the interpolation parameters for a particular line. Old CMISS name XPXE, ZPZE.*

- subroutine **FIELD\_ROUTINES::FIELD\_VARIABLES\_FINALISE** (FIELD, ERR, ERROR,\*)
 

*Finalises the field variables for a field and deallocates all memory.*
- subroutine **FIELD\_ROUTINES::FIELD\_VARIABLES\_INITIALISE** (FIELD, ERR, ERROR,\*)
 

*Initialises the field variables.*
- subroutine **FIELD\_ROUTINES::FIELDS\_FINALISE** (REGION, ERR, ERROR,\*)
 

*Finalises the fields for the given region and deallocates all memory.*
- subroutine **FIELD\_ROUTINES::FIELDS\_INITIALISE** (REGION, ERR, ERROR,\*)
 

*Initialises the fields for the given region.*

## Variables

- INTEGER(INTG), parameter **FIELD\_ROUTINES::FIELD\_INDEPENDENT\_TYPE** = 1
 

*Independent field type.*
- INTEGER(INTG), parameter **FIELD\_ROUTINES::FIELD\_DEPENDENT\_TYPE** = 2
 

*Dependent field type.*
- INTEGER(INTG), parameter **FIELD\_ROUTINES::FIELD\_SCALAR\_DIMENSION\_TYPE** = 1
 

*Scalar field.*
- INTEGER(INTG), parameter **FIELD\_ROUTINES::FIELD\_VECTOR\_DIMENSION\_TYPE** = 2
 

*Vector field.*
- INTEGER(INTG), parameter **FIELD\_ROUTINES::FIELD\_TENSOR\_DIMENSION\_TYPE** = 2
 

*Tensor field.*
- INTEGER(INTG), parameter **FIELD\_ROUTINES::FIELD\_GEOMETRIC\_TYPE** = 1
 

*Geometric field.*
- INTEGER(INTG), parameter **FIELD\_ROUTINES::FIELD\_FIBRE\_TYPE** = 2
 

*Fibre field.*
- INTEGER(INTG), parameter **FIELD\_ROUTINES::FIELD\_GENERAL\_TYPE** = 3
 

*General field.*
- INTEGER(INTG), parameter **FIELD\_ROUTINES::FIELD\_MATERIAL\_TYPE** = 4
 

*Material field.*
- INTEGER(INTG), parameter **FIELD\_ROUTINES::FIELD\_CONSTANT\_INTERPOLATION** = 1
 

*Constant interpolation. One parameter for the field.*
- INTEGER(INTG), parameter **FIELD\_ROUTINES::FIELD\_ELEMENT\_BASED\_INTERPOLATION** = 2
 

*Element based interpolation. Parameters are different in each element.*

- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_NODE_BASED_INTERPOLATION = 3`  
*Node based interpolation. Parameters are nodal based and a basis function is used.*
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_GRID_POINT_BASED_INTERPOLATION = 4`  
*Grid point based interpolation. Parameters are different at each grid point.*
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_GAUSS_POINT_BASED_INTERPOLATION = 5`  
*Gauss point based interpolation. Parameters are different at each Gauss point.*
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_NUMBER_OF_VARIABLE_TYPES = 4`  
*Number of different field variable types possible.*
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_STANDARD_VARIABLE_TYPE = 1`  
*Standard variable type i.e.,  $u$ .*
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_NORMAL_VARIABLE_TYPE = 2`  
*Normal derivative variable type i.e.,  $du/dn$ .*
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_TIME_DERIV1_VARIABLE_TYPE = 3`  
*First time derivative variable type i.e.,  $du/dt$ .*
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_TIME_DERIV2_VARIABLE_TYPE = 4`  
*Second type derivative variable type i.e.,  $d^2u/dt^2$ .*
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_CONSTANT_DOF_TYPE = 1`  
*The dof is from a field variable component with constant interpolation.*
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_ELEMENT_DOF_TYPE = 2`  
*The dof is from a field variable component with element based interpolation.*
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_NODE_DOF_TYPE = 3`  
*The dof is from a field variable component with node based interpolation.*
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_POINT_DOF_TYPE = 4`  
*The dof is from a field variable component with point based interpolation.*
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_NUMBER_OF_SET_TYPES = 99`  
*The maximum number of different parameter sets for a field.*
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_VALUES_SET_TYPE = 1`  
*The parameter set corresponding to the field values.*
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_BOUNDARY_CONDITIONS_SET_TYPE = 2`

*The parameter set corresponding to the field boundary conditions.*

- INTEGER(INTG), parameter **FIELD\_ROUTINES::FIELD\_INITIAL\_CONDITIONS\_SET\_TYPE = 3**

*The parameter set corresponding to the field initial conditions.*

- INTEGER(INTG), parameter **FIELD\_ROUTINES::FIELD\_ANALYTIC\_SET\_TYPE = 4**

*The parameter set corresponding to the field values.*

- INTEGER(INTG), parameter **FIELD\_ROUTINES::FIELD\_NO\_SCALING = 0**

*The field is not scaled.*

- INTEGER(INTG), parameter **FIELD\_ROUTINES::FIELD\_UNIT\_SCALING = 1**

*The field has unit scaling.*

- INTEGER(INTG), parameter **FIELD\_ROUTINES::FIELD\_ARC\_LENGTH\_SCALING = 2**

*The field has arc length scaling.*

- INTEGER(INTG), parameter **FIELD\_ROUTINES::FIELD\_ARITHMETIC\_MEAN\_SCALING = 3**

*The field has arithmetic mean of the arc length scaling.*

- INTEGER(INTG), parameter **FIELD\_ROUTINES::FIELD\_HARMONIC\_MEAN\_SCALING = 4**

*The field has geometric mean of the arc length scaling.*

## 8.29.1 Detailed Description

### Id

**field\_routines.f90** 180 2008-10-26 16:45:43Z chrisbradley

### Author:

Chris Bradley This module handles all field related routines.

## 8.29.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University

of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [field\\_routines.f90](#).

## 8.30 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/finite\_elasticity\_routines.f90 File Reference

### Id

[finite\\_elasticity\\_routines.f90](#) 181 2008-10-26 18:59:43Z chrispbradley

### Namespaces

- namespace [FINITE\\_ELASTICITY\\_ROUTINES](#)

*This module handles all finite elasticity routines.*

### Functions

- subroutine [FINITE\\_ELASTICITY\\_ROUTINES::FINITE\\_ELASTICITYFINITEELEMENTJACOBIAN\\_EVALUATE](#) (EQUATIONS\_SET, ELEMENT\_NUMBER, ERR, ERROR,\*)  
*Evaluates the Jacobian for a finite elasticity finite element equations set.*
- subroutine [FINITE\\_ELASTICITY\\_ROUTINES::FINITE\\_ELASTICITYFINITEELEMENTRESIDUAL\\_EVALUATE](#) (EQUATIONS\_SET, ELEMENT\_NUMBER, ERR, ERROR,\*)  
*Evaluates the residual and RHS vectors for a finite elasticity finite element equations set.*
- subroutine [FINITE\\_ELASTICITY\\_ROUTINES::FINITE\\_ELASTICITYGAUSSDEFORMATIONGRADEINT\\_TENSOR](#) (INTERPOLATED\_POINT\_DEF, INT\_RPOLATED\_POINT\_FIBRE,&INTERPOLATED\_POINT\_UNDEF, ELEMENT\_NUMBER, GAUSS\_PT\_NUMBER, DZDNU, Jxxi, Jznu, ERR, ERROR,\*)  
*Evaluates the deformation gradient tensor at a given Gauss point.*
- subroutine [FINITE\\_ELASTICITY\\_ROUTINES::FINITE\\_ELASTICITYGAUSS\\_DXDNU](#) (INTERPOLATED\_POINT, DXDNU, DXDXI, ERR, ERROR,\*)  
*Evaluates  $dx/dn_u$ (undeformed-material cs) tensor at a given Gauss point.*
- subroutine [FINITE\\_ELASTICITY\\_ROUTINES::FINITE\\_ELASTICITYGAUSS\\_CAUCHY\\_TENSOR](#) (FIELD, INTERPOLATED\_POINT, CAUCHY\_TENSOR, DZDNU, ERR, ERROR,\*)  
*Evaluates the Cauchy stress tensor at a given Gauss point.*
- subroutine [FINITE\\_ELASTICITY\\_ROUTINES::FINITE\\_ELASTICITYGAUSS\\_DFDZ](#) (INTERPOLATED\_POINT, DFDZ, XI, ERR, ERROR,\*)  
*Evaluates  $df/dz$  (derivative of interpolation function wrt deformed coord) matrix at a given Gauss point.*
- subroutine [FINITE\\_ELASTICITY\\_ROUTINES::FINITE\\_ELASTICITYEQUATIONSSET\\_SETUP](#) (EQUATIONS\_SET, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)  
*Sets up the finite elasticity equation type of an elasticity equations set class.*
- subroutine [FINITE\\_ELASTICITY\\_ROUTINES::FINITE\\_ELASTICITYEQUATIONSSET\\_SUBTYPE\\_SET](#) (EQUATIONS\_SET, EQUATIONS\_SET\_SUBTYPE, ERR, ERROR,\*)  
*Sets/changes the equation subtype for a finite elasticity equation type of an elasticity equations set class.*

- subroutine [FINITE\\_ELASTICITY\\_ROUTINES::FINITE\\_ELASTICITY\\_PROBLEM\\_SETUP](#)(PROBLEM, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)

*Sets up the finite elasticity problem.*

- subroutine [FINITE\\_ELASTICITY\\_ROUTINES::FINITE\\_ELASTICITY\\_PROBLEM\\_SUBTYPE\\_SET](#)(PROBLEM, PROBLEM\_SUBTYPE, ERR, ERROR,\*)

*Sets/changes the problem subtype for a finite elasticity type .*

### 8.30.1 Detailed Description

#### Id

[finite\\_elasticity\\_routines.f90](#) 181 2008-10-26 18:59:43Z chrispbradley

#### Author:

Chris Bradley This module handles all finite elasticity routines.

### 8.30.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s): Kumar Mithraratne

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [finite\\_elasticity\\_routines.f90](#).

## **8.31 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/finite\_element\_routines.f90 File Reference**

### **Id**

[finite\\_element\\_routines.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

### **Namespaces**

- namespace [FINITE\\_ELEMENT\\_ROUTINES](#)  
*This module contains all finite element base routines.*

### **Functions**

- subroutine [FINITE\\_ELEMENT\\_ROUTINES::FEM\\_ELEMENT\\_MATRICES\\_FINALISE](#)(ELEMENT\_MATRICES, ERR, ERROR,\*)
- subroutine [FINITE\\_ELEMENT\\_ROUTINES::FEM\\_ELEMENT\\_MATRICES\\_INITIALISE](#)(PROBLEM, ELEMENT\_MATRICES, ERR, ERROR,\*)

#### **8.31.1 Detailed Description**

### **Id**

[finite\\_element\\_routines.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

### **Author:**

Chris Bradley This module contains all finite element base routines.

#### **8.31.2 LICENSE**

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and

not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [finite\\_element\\_routines.f90](#).

## 8.32 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/fluid\_mechanics\_routines.f90 File Reference

### Id

[fluid\\_mechanics\\_routines.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

### Namespaces

- namespace [FLUID\\_MECHANICS\\_ROUTINES](#)

*This module handles all fluid mechanics class routines.*

### 8.32.1 Detailed Description

#### Id

[fluid\\_mechanics\\_routines.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

#### Author:

Chris Bradley This module handles all fluid mechanics routines.

### 8.32.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [fluid\\_mechanics\\_routines.f90](#).

## 8.33 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/generated\_mesh\_routines.f90 File Reference

### Id

[generated\\_mesh\\_routines.f90](#) 178 2008-10-26 13:07:57Z chrisbradley

### Classes

- interface [GENERATED\\_MESH\\_ROUTINES::GENERATED\\_MESH\\_BASIS\\_SET](#)
- interface [GENERATED\\_MESH\\_ROUTINES::GENERATED\\_MESH\\_DESTROY](#)
- interface [GENERATED\\_MESH\\_ROUTINES::GENERATED\\_MESH\\_EXTENT\\_SET](#)
- interface [GENERATED\\_MESH\\_ROUTINES::GENERATED\\_MESH\\_NUMBER\\_OF\\_ELEMENTS\\_SET](#)
- interface [GENERATED\\_MESH\\_ROUTINES::GENERATED\\_MESH\\_ORIGIN\\_SET](#)
- interface [GENERATED\\_MESH\\_ROUTINES::GENERATED\\_MESH\\_TYPE\\_SET](#)

### Namespaces

- namespace [GENERATED\\_MESH\\_ROUTINES](#)

*This module handles all generated mesh routines.*

### Functions

- TYPE(BASIS\_TYPE) [GENERATED\\_MESH\\_ROUTINES::GENERATED\\_MESH\\_BASIS\\_GET](#)(GENERATED\_MESH, ERR, ERROR)
 

*Gets the basis of a generated mesh.*
- subroutine [GENERATED\\_MESH\\_ROUTINES::GENERATED\\_MESH\\_BASIS\\_SET\\_NUMBER](#)(USER\_NUMBER, BASIS, ERR, ERROR,\*)
 

*Set the basis of the generated mesh.*
- subroutine [GENERATED\\_MESH\\_ROUTINES::GENERATED\\_MESH\\_BASIS\\_SET\\_PTR](#)(GENERATED\_MESH, BASIS, ERR, ERROR,\*)
 

*Sets/changes the basis of a generated mesh.*
- subroutine [GENERATED\\_MESH\\_ROUTINES::GENERATED\\_MESH\\_CREATE\\_FINISH](#)(GENERATED\_MESH, MESH, MESH\_USER\_NUMBER, ERR, ERROR,\*)
 

*Finishes the creation of a generated mesh.*
- subroutine [GENERATED\\_MESH\\_ROUTINES::GENERATED\\_MESH\\_CREATE\\_START](#)(USER\_NUMBER, REGION, GENERATED\_MESH, ERR, ERROR,\*)
 

*Starts the creation of a generated mesh.*
- subroutine [GENERATED\\_MESH\\_ROUTINES::GENERATED\\_MESH\\_DESTROY\\_NUMBER](#)(USER\_NUMBER, ERR, ERROR,\*)
 

*Destroys a generated mesh.*

- subroutine **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_DESTROY\_PTR** (GENERATED\_MESH, ERR, ERROR,\*)
 

*Destroys a generated mesh.*
- REAL(DP), pointer **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_EXTENT\_GET** (GENERATED\_MESH, ERR, ERROR)
 

*Gets the extent of a generated mesh.*
- subroutine **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_EXTENT\_SET\_NUMBER** (USER\_NUMBER, MAX\_EXTENT, ERR, ERROR,\*)
 

*Sets/changes the max extent of a generated mesh.*
- subroutine **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_EXTENT\_SET\_PTR** (GENERATED\_MESH, MAX\_EXTENT, ERR, ERROR,\*)
 

*Sets/changes the extent of a generated mesh.*
- subroutine **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_FINALISE** (GENERATED\_MESH, ERR, ERROR,\*)
 

*Finalises a generated mesh and deallocates all memory.*
- subroutine **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_INITIALISE** (GENERATED\_MESH, ERR, ERROR,\*)
 

*Initialises a generated mesh.*
- INTEGER(INTG), pointer **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_NUMBER\_OF\_ELEMENTS\_GET** (GENERATED\_MESH, ERR, ERROR)
 

*Gets the extent of a generated mesh.*
- subroutine **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_NUMBER\_OF\_ELEMENTS\_SET\_NUMBER** (USER\_NUMBER, NUMBER\_OF\_ELEMENTS\_XI, ERR, ERROR,\*)
 

*Sets/changes the extent of a generated mesh.*
- subroutine **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_NUMBER\_OF\_ELEMENTS\_SET\_PTR** (GENERATED\_MESH, NUMBER\_OF\_ELEMENTS\_XI, ERR, ERROR,\*)
 

*Sets/changes the extent of a generated mesh.*
- REAL(DP), pointer **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_ORIGIN\_GET** (GENERATED\_MESH, ERR, ERROR)
 

*Get the origin of a generated mesh.*
- subroutine **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_ORIGIN\_SET\_NUMBER** (USER\_NUMBER, ORIGIN, ERR, ERROR,\*)
 

*Sets/changes the origin of a generated mesh.*
- subroutine **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_ORIGIN\_SET\_PTR** (GENERATED\_MESH, ORIGIN, ERR, ERROR,\*)
 

*Sets/changes the origin of a generated mesh.*
- subroutine **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_REGULAR\_CREATE\_FINISH** (GENERATED\_MESH, MESH\_USER\_NUMBER, ERR, ERROR,\*)
 

*Start to create the regular generated mesh type.*

- subroutine **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_REGULAR\_FINALISE** (REGULAR\_MESH, ERR, ERROR,\*)  
*Finalise the regular mesh type.*
  - subroutine **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_REGULAR\_INITIALISE** (GENERATED\_MESH, ERR, ERROR,\*)  
*Initialise the regular generated mesh type.*
  - INTEGER(INTG) **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_TYPE\_GET** (GENERATED\_MESH, ERR, ERROR)  
*Gets the type of a generated mesh.*
  - subroutine **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_TYPE\_SET\_NUMBER** (USER\_NUMBER, GENERATED\_TYPE, ERR, ERROR,\*)  
*Sets/changes the type of a generated mesh.*
  - subroutine **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_TYPE\_SET\_PTR** (GENERATED\_MESH, GENERATED\_TYPE, ERR, ERROR,\*)  
*Sets/changes the type of a generated mesh.*
  - subroutine **GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_USER\_NUMBER\_FIND** (USER\_NUMBER, GENERATED\_MESH, ERR, ERROR,\*)  
*Finds and returns in generated mesh a pointer to that identified by USER\_NUMBER. If no generated mesh with that number exists left nullified.*
  - subroutine **GENERATED\_MESH\_ROUTINES::GENERATED\_MESHES\_FINALISE** (ERR, ERROR,\*)  
*Finalises all generated meshes and deallocates all memory.*
  - subroutine **GENERATED\_MESH\_ROUTINES::GENERATED\_MESHES\_INITIALISE** (ERR, ERROR,\*)  
*Initialises all generated meshes.*

## Variables

- INTEGER(INTG), parameter **REGULAR\_MESH\_TYPE** = 1  
*A regular generated mesh.*
  - INTEGER(INTG), parameter **POLAR\_MESH\_TYPE** = 2  
*A polar generated mesh.*
  - INTEGER(INTG), parameter **FRACTAL\_TREE\_MESH\_TYPE** = 3  
*A fractal tree generated mesh.*
  - TYPE(GENERATED\_MESHES\_TYPE),  
**ROUTINES::GENERATED\_MESHES** target **GENERATED\_MESH\_**-

### 8.33.1 Detailed Description

#### Id

[generated\\_mesh\\_routines.f90](#) 178 2008-10-26 13:07:57Z chrisbradley

#### Author:

Chris Bradley This module handles all generated mesh routines.

#### Todo

Move generated regular mesh from mesh routines and generalise.

### 8.33.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [generated\\_mesh\\_routines.f90](#).

## 8.34 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/input\_-output.f90 File Reference

### Id

[input\\_output.f90](#) 177 2008-10-25 13:38:18Z chrispb

### Classes

- interface [INPUT\\_OUTPUT::WRITE\\_STRING](#)  
*Write a string to a given output stream.*
- interface [INPUT\\_OUTPUT::WRITE\\_STRING\\_VALUE](#)  
*Write a string followed by a value to a given output stream.*
- interface [INPUT\\_OUTPUT::WRITE\\_STRING\\_TWO\\_VALUE](#)  
*Write a string, value, string then a value to a given output stream.*
- interface [INPUT\\_OUTPUT::WRITE\\_STRING\\_FMT\\_VALUE](#)  
*Write a string followed by a value formatted in a particular way to a specified output stream.*
- interface [INPUT\\_OUTPUT::WRITE\\_STRING\\_FMT\\_TWO\\_VALUE](#)  
*Write a string, value, string then a value with the values formatted in a particular way to a given output stream.*
- interface [INPUT\\_OUTPUT::WRITE\\_STRING\\_VECTOR](#)  
*Write a string followed by a vector to a specified output stream.*
- interface [INPUT\\_OUTPUT::WRITE\\_STRING\\_IDX\\_VECTOR](#)  
*Write a string followed by a indexed vector to a specified output stream.*
- interface [INPUT\\_OUTPUT::WRITE\\_STRING\\_MATRIX](#)  
*Write a string followed by a matrix to a specified output stream.*

### Namespaces

- namespace [INPUT\\_OUTPUT](#)  
*This module handles all formating and input and output.*

### Functions

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_C](#) (ID, STRING, ERR, ERROR,\*)  
*Writes the character STRING to the given output stream specified by ID.*
- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_VS](#) (ID, STRING, ERR, ERROR,\*)  
*Writes the varying string STRING to the given output stream specified by ID.*

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_VALUE\\_C](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)  
*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.*
- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_VALUE\\_DP](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)  
*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.*
- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_VALUE\\_INTG](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)  
*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.*
- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_VALUE\\_LINTG](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)  
*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.*
- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_VALUE\\_L](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)  
*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.*
- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_VALUE\\_SP](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)  
*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.*
- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_VALUE\\_VS](#) (ID, FIRST\_STRING, VALUE, ERR, ERROR,\*)  
*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.*
- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_TWO\\_VALUE\\_C\\_C](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)  
*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_TWO\\_VALUE\\_C\\_DP](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)  
*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_TWO\\_VALUE\\_C\\_INTG](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)  
*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_TWO\\_VALUE\\_C\\_L](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_TWO\\_VALUE\\_C\\_SP](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_TWO\\_VALUE\\_C\\_VS](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted varying string SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_TWO\\_VALUE\\_DP\\_C](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_TWO\\_VALUE\\_DP\\_DP](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_TWO\\_VALUE\\_DP\\_INTG](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_TWO\\_VALUE\\_DP\\_L](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_TWO\\_VALUE\\_DP\\_SP](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_TWO\\_VALUE\\_DP\\_VS](#) (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_C` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_DP` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_INTG` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_L` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_SP` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_VS` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_C` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_DP` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_INTG` (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine **INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_L\_L** (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine **INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_L\_SP** (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine **INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_L\_VS** (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine **INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_SP\_C** (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine **INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_SP\_DP** (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine **INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_SP\_INTG** (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine **INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_SP\_L** (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine **INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_SP\_SP** (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*
- subroutine **INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_SP\_VS** (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine **INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_VS\_C** (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine **INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_VS\_DP** (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine **INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_VS\_INTG** (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine **INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_VS\_L** (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine **INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_VS\_SP** (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine **INPUT\_OUTPUT::WRITE\_STRING\_TWO\_VALUE\_VS\_VS** (ID, FIRST\_STRING, FIRST\_VALUE, SECOND\_STRING, SECOND\_VALUE, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted varying string SECOND\_VALUE to the given output stream specified by ID. Free format is used to format both values.*

- subroutine **INPUT\_OUTPUT::WRITE\_STRING\_FMT\_VALUE\_C** (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)

*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.*

- subroutine **INPUT\_OUTPUT::WRITE\_STRING\_FMT\_VALUE\_DP** (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)

*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.*

- subroutine **INPUT\_OUTPUT::WRITE\_STRING\_FMT\_VALUE\_INTG** (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)

*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.*

- subroutine `INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_LINTG` (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)
 

*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.*
- subroutine `INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_L` (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)
 

*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.*
- subroutine `INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_SP` (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)
 

*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.*
- subroutine `INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_VS` (ID, FIRST\_STRING, VALUE, FORMAT\_STRING, ERR, ERROR,\*)
 

*Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT\_STRING is used to format the value.*
- subroutine `INPUT_OUTPUT::WR` (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*
- subroutine `INPUT_OUTPUT::WR` (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formmatted double precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*
- subroutine `INPUT_OUTPUT::WR` (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formmatted integer SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*
- subroutine `INPUT_OUTPUT::WR` (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formmatted logical SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*
- subroutine `INPUT_OUTPUT::WR` (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)
 

*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formmatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **INPUT\_OUTPUT::WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted character FIRST\_VALUE and the the SECOND\_STRING followed by a formatted varying string SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **INPUT\_OUTPUT::WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **INPUT\_OUTPUT::WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **INPUT\_OUTPUT::WRITE\_STRING\_FMT** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE,&SECOND\_FORMAT, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **INPUT\_OUTPUT::WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **INPUT\_OUTPUT::WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **INPUT\_OUTPUT::WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted double precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **INPUT\_OUTPUT::WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_FMT](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE,&SECOND\_FORMAT, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_FMT](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE,&SECOND\_FORMAT, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [INPUT\\_OUTPUT::WR](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_FMT](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE,&SECOND\_FORMAT, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_FMT](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE,&SECOND\_FORMAT, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted integer FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [INPUT\\_OUTPUT::WR](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [INPUT\\_OUTPUT::WR](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **INPUT\_OUTPUT::WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **INPUT\_OUTPUT::WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **INPUT\_OUTPUT::WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **INPUT\_OUTPUT::WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted logical FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **INPUT\_OUTPUT::WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **INPUT\_OUTPUT::WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formmatted double precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **INPUT\_OUTPUT::WRITE\_STRING\_FMT** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE,&SECOND\_FORMAT, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formmatted integer SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine **INPUT\_OUTPUT::WR** (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formmatted logical SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [\*\*INPUT\\_OUTPUT::WR\*\*](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [\*\*INPUT\\_OUTPUT::WR\*\*](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted single precision FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [\*\*INPUT\\_OUTPUT::WR\*\*](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted character SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [\*\*INPUT\\_OUTPUT::WR\*\*](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted double precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [\*\*INPUT\\_OUTPUT::WRITE\\_STRING\\_FMT\*\*](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE,&SECOND\_FORMAT, ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted integer SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [\*\*INPUT\\_OUTPUT::WR\*\*](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted logical SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [\*\*INPUT\\_OUTPUT::WR\*\*](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted single precision SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [\*\*INPUT\\_OUTPUT::WR\*\*](#) (ID, FIRST\_STRING, FIRST\_VALUE, FIRST\_FORMAT, SECOND\_STRING, SECOND\_VALUE, SECOND\_FORMAT,&ERR, ERROR,\*)

*Writes the FIRST\_STRING followed by a formatted varying string FIRST\_VALUE and the the SECOND\_STRING followed by a formatted varying string SECOND\_VALUE to the given output stream specified by ID. FIRST\_FORMAT is used to format the first value and SECOND\_FORMAT is used to format the second value.*

- subroutine [INPUT\\_OUTPUT::WR](#) (ID, FIRST\_IDX, DELTA, LAST\_IDX, NUMBER\_FIRST, NUMBER\_REPEAT, VECTOR, FIRST\_FORMAT, REPEAT\_FORMAT,&ERR, ERROR,\*)

*Writes the given double precision VECTOR to the given output stream specified by ID. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_IDX and LAST\_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.*

- subroutine [INPUT\\_OUTPUT::WR](#) (ID, FIRST\_IDX, DELTA, LAST\_IDX, NUMBER\_FIRST, NUMBER\_REPEAT, VECTOR, FIRST\_FORMAT, REPEAT\_FORMAT,&ERR, ERROR,\*)

*Writes the given integer VECTOR to the given output stream specified by ID. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_IDX and LAST\_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.*

- subroutine [INPUT\\_OUTPUT::WR](#) (ID, FIRST\_IDX, DELTA, LAST\_IDX, NUMBER\_FIRST, NUMBER\_REPEAT, VECTOR, FIRST\_FORMAT, REPEAT\_FORMAT,&ERR, ERROR,\*)

*Writes the given integer VECTOR to the given output stream specified by ID. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_IDX and LAST\_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.*

- subroutine [INPUT\\_OUTPUT::WR](#) (ID, FIRST\_IDX, DELTA, LAST\_IDX, NUMBER\_FIRST, NUMBER\_REPEAT, VECTOR, FIRST\_FORMAT, REPEAT\_FORMAT,&ERR, ERROR,\*)

*Writes the given logical VECTOR to the given output stream specified by ID. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_IDX and LAST\_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.*

- subroutine [INPUT\\_OUTPUT::WR](#) (ID, FIRST\_IDX, DELTA, LAST\_IDX, NUMBER\_FIRST, NUMBER\_REPEAT, VECTOR, FIRST\_FORMAT, REPEAT\_FORMAT,&ERR, ERROR,\*)

*Writes the given single precision VECTOR to the given output stream specified by ID. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_IDX and LAST\_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.*

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_IDX](#) (ID, NUM\_INDICES, INDICES, DELTA, NUMBER\_FIRST, NUMBER\_REPEAT, VECTOR, FIRST\_FORMAT,&REPEAT\_FORMAT, ERR, ERROR,\*)

*Writes the given indexed double precision VECTOR to the given output stream specified by ID. NUM\_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. DELTA is the number of actual indices to skip for each index.*

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_IDX](#) (ID, NUM\_INDICES, INDICES, DELTA, NUMBER\_FIRST, NUMBER\_REPEAT, VECTOR, FIRST\_FORMAT,&REPEAT\_FORMAT, ERR, ERROR,\*)

*Writes the given indexed integer VECTOR to the given output stream specified by ID. NUM\_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. DELTA is the number of actual indices to skip for each index.*

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_IDX](#) (ID, NUM\_INDICES, INDICES, DELTA, NUMBER\_FIRST, NUMBER\_REPEAT, VECTOR, FIRST\_FORMAT,&REPEAT\_FORMAT, ERR, ERROR,\*)

*Writes the given indexed integer VECTOR to the given output stream specified by ID. NUM\_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. DELTA is the number of actual indices to skip for each index.*

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_IDX](#) (ID, NUM\_INDICES, INDICES, DELTA, NUMBER\_FIRST, NUMBER\_REPEAT, VECTOR, FIRST\_FORMAT,&REPEAT\_FORMAT, ERR, ERROR,\*)

*Writes the given indexed logical VECTOR to the given output stream specified by ID. NUM\_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. DELTA is the number of actual indices to skip for each index.*

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_IDX](#) (ID, NUM\_INDICES, INDICES, DELTA, NUMBER\_FIRST, NUMBER\_REPEAT, VECTOR, FIRST\_FORMAT,&REPEAT\_FORMAT, ERR, ERROR,\*)

*Writes the given indexed single precision VECTOR to the given output stream specified by ID. NUM\_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. DELTA is the number of actual indices to skip for each index.*

- subroutine [INPUT\\_OUTPUT::WRITE\\_STRING\\_MATRIX\\_DP](#) (ID, FIRST\_ROW, DELTA\_ROW, LAST\_ROW, FIRST\_COLUMN, DELTA\_COLUMN, LAST\_COLUMN, NUMBER\_FIRS,&NUMBER\_REPEAT, MATRIX, INDEX\_FORMAT\_TYPE, MATRIX\_NAME\_FORMAT, ROW\_INDEX\_FORMAT, FIRST\_FORMAT, REPEAT\_FORMAT, ERR, ERROR,\*)

*Writes the given double precision MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX\_FORMAT\_TYPE. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_ONLY then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated named with the FIRST\_FORMAT. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated with ROW\_INDEX\_FORMAT and concatenated with FIRST\_FORMAT. Note that with a WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_ROW/FIRST\_COLUMN and LAST\_ROW/LAST\_COLUMN are the extents of the row/column and DELTA\_ROW/DELTA\_COLUMN is the NUMBER of indices to skip for each row/column index.*

- subroutine `INPUT_OUTPUT::WRITE_STRING_MATRIX_INTG` (ID, FIRST\_ROW, DELTA\_ROW, LAST\_ROW, FIRST\_COLUMN, DELTA\_COLUMN, LAST\_COLUMN, NUMBER\_FIRST,&NUMBER\_REPEAT, MATRIX, INDEX\_FORMAT\_TYPE, MATRIX\_NAME\_FORMAT, ROW\_INDEX\_FORMAT, FIRST\_FORMAT, REPEAT\_FORMAT, ERR, ERROR,\*)

*Writes the given integer MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX\_FORMAT\_TYPE. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_ONLY then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated named with the FIRST\_FORMAT. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated with ROW\_INDEX\_FORMAT and concatenated with FIRST\_FORMAT. Note that with a WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_ROW/FIRST\_COLUMN and LAST\_ROW/LAST\_COLUMN are the extents of the row/column and DELTA\_ROW/DELTA\_COLUMN is the NUMBER of indices to skip for each row/column index.*

- subroutine `INPUT_OUTPUT::WRITE_STRING_MATRIX_LINTG` (ID, FIRST\_ROW, DELTA\_ROW, LAST\_ROW, FIRST\_COLUMN, DELTA\_COLUMN, LAST\_COLUMN, NUMBER\_FIRST,&NUMBER\_REPEAT, MATRIX, INDEX\_FORMAT\_TYPE, MATRIX\_NAME\_FORMAT, ROW\_INDEX\_FORMAT, FIRST\_FORMAT, REPEAT\_FORMAT, ERR, ERROR,\*)
- subroutine `INPUT_OUTPUT::WRITE_STRING_MATRIX_L` (ID, FIRST\_ROW, DELTA\_ROW, LAST\_ROW, FIRST\_COLUMN, DELTA\_COLUMN, LAST\_COLUMN, NUMBER\_FIRST &NUMBER\_REPEAT, MATRIX, INDEX\_FORMAT\_TYPE, MATRIX\_NAME\_FORMAT, ROW\_INDEX\_FORMAT, FIRST\_FORMAT, REPEAT\_FORMAT, ERR, ERROR,\*)

*Writes the given logical MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX\_FORMAT\_TYPE. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_ONLY then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated named with the FIRST\_FORMAT. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated with ROW\_INDEX\_FORMAT and concatenated with FIRST\_FORMAT. Note that with a WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_ROW/FIRST\_COLUMN and LAST\_ROW/LAST\_COLUMN are the extents of the row/column and DELTA\_ROW/DELTA\_COLUMN is the NUMBER of indices to skip for each row/column index.*

- subroutine `INPUT_OUTPUT::WRITE_STRING_MATRIX_SP` (ID, FIRST\_ROW, DELTA\_ROW, LAST\_ROW, FIRST\_COLUMN, DELTA\_COLUMN, LAST\_COLUMN, NUMBER\_FIRST,&NUMBER\_REPEAT, MATRIX, INDEX\_FORMAT\_TYPE, MATRIX\_NAME\_FORMAT, ROW\_INDEX\_FORMAT, FIRST\_FORMAT, REPEAT\_FORMAT, ERR, ERROR,\*)

*Writes the given single precision MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX\_FORMAT\_TYPE. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_ONLY then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated named with the FIRST\_FORMAT. If INDEX\_FORMAT\_TYPE is WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES then the first line of output for each row is MATRIX\_NAME\_FORMAT concatenated with ROW\_INDEX\_FORMAT and concatenated with FIRST\_FORMAT. Note that with a WRITE\_STRING\_MATRIX\_NAME\_AND\_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST\_FORMAT is the format initially used, followed by the REPEAT\_FORMAT which is repeated as many times as necessary. NUMBER\_FIRST is the number of data items in the FIRST\_FORMAT and NUMBER\_REPEAT is the number of data items in the REPEAT\_FORMAT. FIRST\_ROW/FIRST\_COLUMN and LAST\_ROW/LAST\_COLUMN are the extents of the row/column and DELTA\_ROW/DELTA\_COLUMN is the NUMBER of indices to skip for each row/column index.*

## Variables

- INTEGER(INTG), parameter `INPUT_OUTPUT::WRITE_STRING_MATRIX_NAME_ONLY = 1`

*Write the matrix name with out any indices.*

- INTEGER(INTG), parameter `INPUT_OUTPUT::WRITE_STRING_MATRIX_NAME_AND_INDICES = 2`

*Write the matrix name together with the matrix indices.*

### 8.34.1 Detailed Description

#### Id

`input_output.f90` 177 2008-10-25 13:38:18Z chrispb Bradley

#### Author:

Chris Bradley This module handles all formating and input and output.

### 8.34.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file `input_output.f90`.

## **8.35 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/iso\_varying\_string.f90 File Reference**

### **Id**

[iso\\_varying\\_string.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

### **Classes**

- struct [ISO\\_VARYING\\_STRING::VARYING\\_STRING](#)
- interface [ISO\\_VARYING\\_STRING::assignment\(=\)](#)
- interface [ISO\\_VARYING\\_STRING::adjustr](#)
- interface [ISO\\_VARYING\\_STRING::char](#)
- interface [ISO\\_VARYING\\_STRING::iachar](#)
- interface [ISO\\_VARYING\\_STRING::ichar](#)
- interface [ISO\\_VARYING\\_STRING::index](#)
- interface [ISO\\_VARYING\\_STRING::len](#)
- interface [ISO\\_VARYING\\_STRING::len\\_trim](#)
- interface [ISO\\_VARYING\\_STRING::lge](#)
- interface [ISO\\_VARYING\\_STRING::lgt](#)
- interface [ISO\\_VARYING\\_STRING::lle](#)
- interface [ISO\\_VARYING\\_STRING::llt](#)
- interface [ISO\\_VARYING\\_STRING::repeat](#)
- interface [ISO\\_VARYING\\_STRING::scan](#)
- interface [ISO\\_VARYING\\_STRING::trim](#)
- interface [ISO\\_VARYING\\_STRING::verify](#)
- interface [ISO\\_VARYING\\_STRING::var\\_str](#)
- interface [ISO\\_VARYING\\_STRING::get](#)
- interface [ISO\\_VARYING\\_STRING::put](#)
- interface [ISO\\_VARYING\\_STRING::put\\_line](#)
- interface [ISO\\_VARYING\\_STRING::extract](#)
- interface [ISO\\_VARYING\\_STRING::insert](#)
- interface [ISO\\_VARYING\\_STRING::remove](#)
- interface [ISO\\_VARYING\\_STRING::replace](#)
- interface [ISO\\_VARYING\\_STRING::split](#)

### **Namespaces**

- namespace [ISO\\_VARYING\\_STRING](#)

*This module provides an iso\_varying\_string module, conformant to the API specified in ISO/IEC 1539-2:2000 (varying-length strings for Fortran 95).*

### **Functions**

- integer [ISO\\_VARYING\\_STRING::len\\_\(string\)](#)
- subroutine [ISO\\_VARYING\\_STRING::op\\_assign\\_CH\\_VS](#) (var, exp)
- subroutine [ISO\\_VARYING\\_STRING::op\\_assign\\_VS\\_CH](#) (var, exp)
- type(varying\_string) [ISO\\_VARYING\\_STRING::op\\_concat\\_VS\\_VS](#) (string\_a, string\_b)

- type(varying\_string) ISO\_VARYING\_STRING::op\_concat\_CH\_VS (string\_a, string\_b)
- type(varying\_string) ISO\_VARYING\_STRING::op\_concat\_VS\_CH (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::op\_eq\_VS\_VS (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::op\_eq\_CH\_VS (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::op\_eq\_VS\_CH (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::op\_ne\_VS\_VS (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::op\_ne\_CH\_VS (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::op\_ne\_VS\_CH (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::op\_lt\_VS\_VS (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::op\_lt\_CH\_VS (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::op\_lt\_VS\_CH (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::op\_le\_VS\_VS (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::op\_le\_CH\_VS (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::op\_le\_VS\_CH (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::op\_ge\_VS\_VS (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::op\_ge\_CH\_VS (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::op\_ge\_VS\_CH (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::op\_gt\_VS\_VS (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::op\_gt\_CH\_VS (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::op\_gt\_VS\_CH (string\_a, string\_b)
- type(varying\_string) ISO\_VARYING\_STRING::adjustl\_ (string)
- type(varying\_string) ISO\_VARYING\_STRING::adjustr\_ (string)
- function ISO\_VARYING\_STRING::char\_auto (string)
- character(LEN=length) ISO\_VARYING\_STRING::char\_fixed (string, length)
- integer ISO\_VARYING\_STRING::iachar\_ (c)
- integer ISO\_VARYING\_STRING::ichar\_ (c)
- integer ISO\_VARYING\_STRING::index\_VS\_VS (string, substring, back)
- integer ISO\_VARYING\_STRING::index\_CH\_VS (string, substring, back)
- integer ISO\_VARYING\_STRING::index\_VS\_CH (string, substring, back)
- integer ISO\_VARYING\_STRING::len\_trim\_ (string)
- logical ISO\_VARYING\_STRING::lge\_VS\_VS (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::lge\_CH\_VS (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::lge\_VS\_CH (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::lgt\_VS\_VS (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::lgt\_CH\_VS (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::lgt\_VS\_CH (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::lle\_VS\_VS (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::lle\_CH\_VS (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::lle\_VS\_CH (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::llt\_VS\_VS (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::llt\_CH\_VS (string\_a, string\_b)
- logical ISO\_VARYING\_STRING::llt\_VS\_CH (string\_a, string\_b)
- type(varying\_string) ISO\_VARYING\_STRING::repeat\_ (string, ncopies)
- integer ISO\_VARYING\_STRING::scan\_VS\_VS (string, set, back)
- integer ISO\_VARYING\_STRING::scan\_CH\_VS (string, set, back)
- integer ISO\_VARYING\_STRING::scan\_VS\_CH (string, set, back)
- type(varying\_string) ISO\_VARYING\_STRING::trim\_ (string)
- integer ISO\_VARYING\_STRING::verify\_VS\_VS (string, set, back)
- integer ISO\_VARYING\_STRING::verify\_CH\_VS (string, set, back)
- integer ISO\_VARYING\_STRING::verify\_VS\_CH (string, set, back)

- type(varying\_string) `ISO_VARYING_STRING::var_str_` (char)
- subroutine `ISO_VARYING_STRING::get_` (string, maxlen, iostat)
- subroutine `ISO_VARYING_STRING::get_unit` (unit, string, maxlen, iostat)
- subroutine `ISO_VARYING_STRING::get_set_VS` (string, set, separator, maxlen, iostat)
- subroutine `ISO_VARYING_STRING::get_set_CH` (string, set, separator, maxlen, iostat)
- subroutine `ISO_VARYING_STRING::get_unit_set_VS` (unit, string, set, separator, maxlen, iostat)
- subroutine `ISO_VARYING_STRING::get_unit_set_CH` (unit, string, set, separator, maxlen, iostat)
- subroutine `ISO_VARYING_STRING::put_VS` (string, iostat)
- subroutine `ISO_VARYING_STRING::put_CH` (string, iostat)
- subroutine `ISO_VARYING_STRING::put_unit_VS` (unit, string, iostat)
- subroutine `ISO_VARYING_STRING::put_unit_CH` (unit, string, iostat)
- subroutine `ISO_VARYING_STRING::put_line_VS` (string, iostat)
- subroutine `ISO_VARYING_STRING::put_line_CH` (string, iostat)
- subroutine `ISO_VARYING_STRING::put_line_unit_VS` (unit, string, iostat)
- subroutine `ISO_VARYING_STRING::put_line_unit_CH` (unit, string, iostat)
- type(varying\_string) `ISO_VARYING_STRING::extract_VS` (string, start, finish)
- type(varying\_string) `ISO_VARYING_STRING::extract_CH` (string, start, finish)
- type(varying\_string) `ISO_VARYING_STRING::insert_VS_VS` (string, start, substring)
- type(varying\_string) `ISO_VARYING_STRING::insert_CH_VS` (string, start, substring)
- type(varying\_string) `ISO_VARYING_STRING::insert_VS_CH` (string, start, substring)
- type(varying\_string) `ISO_VARYING_STRING::insert_CH_CH` (string, start, substring)
- TYPE(varying\_string) `ISO_VARYING_STRING::remove_VS` (string, start, finish)
- type(varying\_string) `ISO_VARYING_STRING::remove_CH` (string, start, finish)
- type(varying\_string) `ISO_VARYING_STRING::replace_VS_VS_auto` (string, start, substring)
- type(varying\_string) `ISO_VARYING_STRING::replace_CH_VS_auto` (string, start, substring)
- type(varying\_string) `ISO_VARYING_STRING::replace_VS_CH_auto` (string, start, substring)
- type(varying\_string) `ISO_VARYING_STRING::replace_CH_CH_auto` (string, start, substring)
- type(varying\_string) `ISO_VARYING_STRING::replace_VS_VS_fixed` (string, start, finish, substring)
- type(varying\_string) `ISO_VARYING_STRING::replace_CH_VS_fixed` (string, start, finish, substring)
- type(varying\_string) `ISO_VARYING_STRING::replace_VS_CH_fixed` (string, start, finish, substring)
- type(varying\_string) `ISO_VARYING_STRING::replace_CH_CH_fixed` (string, start, finish, substring)
- type(varying\_string) `ISO_VARYING_STRING::replace_VS_VS_VS_target` (string, target, substring, every, back)
- type(varying\_string) `ISO_VARYING_STRING::replace_CH_VS_VS_target` (string, target, substring, every, back)
- type(varying\_string) `ISO_VARYING_STRING::replace_VS_CH_VS_target` (string, target, substring, every, back)
- type(varying\_string) `ISO_VARYING_STRING::replace_CH_CH_VS_target` (string, target, substring, every, back)
- type(varying\_string) `ISO_VARYING_STRING::replace_VS_VS_CH_target` (string, target, substring, every, back)
- type(varying\_string) `ISO_VARYING_STRING::replace_CH_VS_CH_target` (string, target, substring, every, back)
- type(varying\_string) `ISO_VARYING_STRING::replace_VS_CH_CH_target` (string, target, substring, every, back)
- type(varying\_string) `ISO_VARYING_STRING::replace_CH_CH_CH_target` (string, target, substring, every, back)
- subroutine `ISO_VARYING_STRING::split_VS` (string, word, set, separator, back)
- subroutine `ISO_VARYING_STRING::split_CH` (string, word, set, separator, back)

## Variables

- `integer`, parameter `ISO_VARYING_STRING::GET_BUFFER_LEN = 256`

### 8.35.1 Detailed Description

#### Id

`iso_varying_string.f90` 177 2008-10-25 13:38:18Z chrispbradley

#### Author:

Rich Townsend <`rhdt@bartol.udel.edu`>

Definition in file `iso_varying_string.f90`.

## 8.36 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/kinds.f90 File Reference

### Id

[kinds.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

### Namespaces

- namespace [KINDS](#)

*This module contains all kind definitions.*

### Variables

- INTEGER, parameter [KINDS::INTG](#) = SELECTED\_INT\_KIND(9)  
*Standard integer kind.*
- INTEGER, parameter [KINDS::SINTG](#) = SELECTED\_INT\_KIND(4)  
*Short integer kind.*
- INTEGER, parameter [KINDS::LINTG](#) = SELECTED\_INT\_KIND(18)  
*Long integer kind.*
- INTEGER, parameter [KINDS::PTR](#) = INTG  
*Pointer integer kind.*
- INTEGER, parameter [KINDS::SP](#) = SELECTED\_REAL\_KIND(6)  
*Single precision real kind.*
- INTEGER, parameter [KINDS::DP](#) = SELECTED\_REAL\_KIND(15)  
*Double precision real kind.*
- INTEGER, parameter [KINDS::QP](#) = SELECTED\_REAL\_KIND(30)  
*Quadruple precision real kind.*
- INTEGER, parameter [KINDS::SPC](#) = KIND((1.0\_SP)
- INTEGER, parameter [KINDS::\\_SP](#)  
*Single precision complex kind.*
- INTEGER, parameter [KINDS::DPC](#) = KIND((1.0\_DP)
- INTEGER, parameter [KINDS::\\_DP](#)  
*Double precision complex kind.*

### 8.36.1 Detailed Description

#### Id

[kinds.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

#### Author:

Chris Bradley This module contains all kind definitions.

### 8.36.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [kinds.f90](#).

## 8.37 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/lapack.f90 File Reference

### Id

[lapack.f90](#) 177 2008-10-25 13:38:18Z chrispb Bradley

### Classes

- interface [LAPACK::interface](#)

### Namespaces

- namespace [LAPACK](#)

*This module contains the interface descriptions to the LAPACK routines.*

#### 8.37.1 Detailed Description

### Id

[lapack.f90](#) 177 2008-10-25 13:38:18Z chrispb Bradley

### Author:

Chris Bradley This module contains the interface descriptions to the LAPACK routines.

#### 8.37.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL

or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [lapack.f90](#).

## 8.38 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/Laplace\_equations\_routines.f90 File Reference

### Id

[Laplace\\_equations\\_routines.f90](#) 178 2008-10-26 13:07:57Z chrisbradley

### Namespaces

- namespace [LAPLACE\\_EQUATIONS\\_ROUTINES](#)

*This module handles all Laplace equations routines.*

### Functions

- subroutine [LAPLACE\\_EQUATIONS\\_ROUTINES::LAPLACE\\_EQUATION\\_ANALYTIC\\_CALCULATE](#) (FIELD, EQUATION\_NUMBER, GEOMETRIC\_PARAMETERS, VARIABLE\_NUMBER, COMPONENT\_NUMBER, NODE\_NUMBER, DERIV\_NUMBER, VALUE, ERR, ERROR,\*)

*Calculates the element stiffness matrices and RHS for a Laplace equation finite element equations set.*

- subroutine [LAPLACE\\_EQUATIONS\\_ROUTINES::LAPLACE\\_EQUATION\\_INTE](#) (FIELD, NODE\_NUMBER, COMPONENT\_NUMBER, VARIABLE\_NUMBER, NUMBER\_OF\_DIMENSIONS,&INTERPOLATED\_POINT, ERR, ERROR,\*)

*Gets the interpolated points for the particular node. Assume the node allocation for an element is fixed.*

- subroutine [LAPLACE\\_EQUATIONS\\_ROUTINES::LAPLACE\\_EQUATION\\_FIXED\\_CONDITIONS\\_](#) (EQUATIONS\_SET, SET\_TYPE, DERIVATIVE\_NUMBER, NODE\_NUMBER, COMPONENT\_NUMBER,&VARIABLE\_NUMBER, CONDITION, VALUE, ERR, ERROR,\*)

*Sets a fixed condition for the equation set on the specified node. This is more generic method, move to other place.*

- subroutine [LAPLACE\\_EQUATIONS\\_ROUTINES::LAPLACE\\_EQUATION\\_ANALYTIC\\_PARAMETER\\_SET\\_UPDATE](#) (EQUATIONS\_SET, GEOMETRIC\_PARAMETERS, NODE\_TYPE, ERR, ERROR,\*)

*Calculates the element stiffness matrices and RHS for a Laplace equation finite element equations set.*

- subroutine [LAPLACE\\_EQUATIONS\\_ROUTINES::LAPLACE\\_EQUATIONFINITE\\_ELEMENT\\_CALCULATE](#) (EQUATIONS\_SET, ELEMENT\_NUMBER, ERR, ERROR,\*)

*Calculates the element stiffness matrices and RHS for a Laplace equation finite element equations set.*

- subroutine [LAPLACE\\_EQUATIONS\\_ROUTINES::LAPLACE\\_EQUATION\\_EQUATIONS\\_SET\\_SETUP](#) (EQUATIONS\_SET, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)

*Sets up the Laplace equation type of a classical field equations set class.*

- subroutine [LAPLACE\\_EQUATIONS\\_ROUTINES::LAPLACE\\_EQUATION\\_EQUATIONS\\_SET\\_SUBTYPE\\_SET](#) (EQUATIONS\_SET, EQUATIONS\_SET\_SUBTYPE, ERR, ERROR,\*)

*Sets/changes the equation subtype for a Laplace equation type of a classical field equations set class.*

- subroutine `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_-SET_STANDARD_SETUP` (`EQUATIONS_SET`, `SETUP_TYPE`, `ACTION_TYPE`, `ERR`, `ERROR,*`)  
*Sets up the standard Laplace equation.*
- subroutine `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SETUP` (`PROBLEM`, `SETUP_TYPE`, `ACTION_TYPE`, `ERR`, `ERROR,*`)  
*Sets up the Laplace solution.*
- subroutine `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_-SUBTYPE_SET` (`PROBLEM`, `PROBLEM_SUBTYPE`, `ERR`, `ERROR,*`)  
*Sets/changes the problem subtype for a Laplace equation type .*
- subroutine `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_-STANDARD_SETUP` (`PROBLEM`, `SETUP_TYPE`, `ACTION_TYPE`, `ERR`, `ERROR,*`)  
*Sets up the standard Laplace equations solution.*

## Variables

- INTEGER(INTG), parameter `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-TWO_DIM_1 = 1`  

$$u=x^{**2}+2*x*y-y^{**2}$$
- INTEGER(INTG), parameter `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-TWO_DIM_2 = 2`  

$$u=\cos(x)\cosh(y)$$
- INTEGER(INTG), parameter `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-THREE_DIM_1 = 3`  

$$u=x^{**2}-2*y^{**2}+z^{**2}$$
- INTEGER(INTG), parameter `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-THREE_DIM_2 = 4`  

$$u=\cos(x)*\cosh(y)*z$$

### 8.38.1 Detailed Description

#### Id

`Laplace_equations_routines.f90` 178 2008-10-26 13:07:57Z chrisbradley

#### Author:

Chris Bradley This module handles all Laplace equations routines.

### 8.38.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [Laplace\\_equations\\_routines.f90](#).

## 8.39 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/linear\_elasticity\_routines.f90 File Reference

### Id

[linear\\_elasticity\\_routines.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

### Namespaces

- namespace [LINEAR\\_ELASTICITY\\_ROUTINES](#)

*This module handles all linear elasticity routines.*

### Functions

- subroutine [LINEAR\\_ELASTICITY\\_ROUTINES::LINEAR\\_ELASTICITYFINITEELEMENTCALCULATE](#) (EQUATIONS\_SET, ELEMENT\_NUMBER, ERR, ERROR,\*)
 

*Calculates the element stiffness matrices and RHS for a linear elasticity finite element equations set.*
- subroutine [LINEAR\\_ELASTICITY\\_ROUTINES::LINEARELASTICITYEQUATIONSSETSETUP](#) (EQUATIONS\_SET, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)
 

*Sets up the Linear elasticity equation type of an elasticity equations set class.*
- subroutine [LINEAR\\_ELASTICITY\\_ROUTINES::LINEARELASTICITYEQUATIONSSETSUBTYPESET](#) (EQUATIONS\_SET, EQUATIONS\_SET\_SUBTYPE, ERR, ERROR,\*)
 

*Sets/changes the equation subtype for a linear elasticity equation type of an elasticity equations set class.*
- subroutine [LINEAR\\_ELASTICITY\\_ROUTINES::LINEARELASTICITYPROBLEMSETUP](#) (PROBLEM, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)
 

*Sets up the linear elasticity problem.*
- subroutine [LINEAR\\_ELASTICITY\\_ROUTINES::LINEARELASTICITYPROBLEMSUBTYPESET](#) (PROBLEM, PROBLEM\_SUBTYPE, ERR, ERROR,\*)
 

*Sets/changes the problem subtype for a linear elasticity type .*

### 8.39.1 Detailed Description

#### Id

[linear\\_elasticity\\_routines.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

#### Author:

Chris Bradley This module handles all linear elasticity routines.

### 8.39.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [linear\\_elasticity\\_routines.f90](#).

## 8.40 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/lists.f90 File Reference

### Id

[lists.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

### Classes

- struct [LISTS::LIST\\_PTR\\_TYPE](#)  
*Buffer type to allow arrays of pointers to a list.*
- struct [LISTS::LIST\\_TYPE](#)  
*Contains information on a list.*
- interface [LISTS::LIST\\_ITEM\\_ADD](#)  
*Adds an item to the end of a list.*
- interface [LISTS::LIST\\_ITEM\\_IN\\_LIST](#)  
*Determines if an item is in a list and returns the position of the item.*
- interface [LISTS::LIST\\_DETACH\\_AND\\_DESTROY](#)  
*Detaches the list values from a list and returns them as a pointer to a array of base type before destroying the list.*
- interface [LISTS::LIST\\_SEARCH](#)  
*Searches a list for a given value and returns the position in the list if the value exists.*
- interface [LISTS::LIST\\_SEARCH\\_LINEAR](#)  
*Searches a list using the linear search method.*
- interface [LISTS::LIST\\_SORT](#)  
*Sorts a list into ascending order.*
- interface [LISTS::LIST\\_SORT\\_BUBBLE](#)  
*Sorts a list into assending order using the bubble sort method.*
- interface [LISTS::LIST\\_SORT\\_HEAP](#)  
*Sorts a list into assending order using the heap sort method.*
- interface [LISTS::LIST\\_SORT\\_SHELL](#)  
*Sorts a list into either assending or descending order using the shell sort method.*

### Namespaces

- namespace [LISTS](#)  
*Implements lists of base types.*

## Functions

- subroutine [LISTS::LIST\\_CREATE\\_FINISH](#) (LIST, ERR, ERROR,\*)
 

*Finishes the creation of a list created with LIST\_CREATE\_START.*
- subroutine [LISTS::LIST\\_CREATE\\_START](#) (LIST, ERR, ERROR,\*)
 

*Starts the creation of a list and returns a pointer to the created list.*
- subroutine [LISTS::LIST\\_DATA\\_TYPE\\_SET](#) (LIST, DATA\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the data type for a list.*
- subroutine [LISTS::LIST\\_DESTROY](#) (LIST, ERR, ERROR,\*)
 

*Destroys a list.*
- subroutine [LISTS::LIST\\_FINALISE](#) (LIST, ERR, ERROR,\*)
 

*Finalises a list and deallocates all memory.*
- subroutine [LISTS::LIST\\_INITIALISE](#) (LIST, ERR, ERROR,\*)
 

*Initialises a list and all its components.*
- subroutine [LISTS::LIST\\_INITIAL\\_SIZE\\_SET](#) (LIST, INITIAL\_SIZE, ERR, ERROR,\*)
 

*Sets/changes the initial size for a list.*
- subroutine [LISTS::LIST\\_ITEM\\_ADD\\_INTG1](#) (LIST, ITEM, ERR, ERROR,\*)
 

*Adds an item to the end of an integer list.*
- subroutine [LISTS::LIST\\_ITEM\\_ADD\\_SP1](#) (LIST, ITEM, ERR, ERROR,\*)
 

*Adds an item to the end of a single precision real list.*
- subroutine [LISTS::LIST\\_ITEM\\_ADD\\_DP1](#) (LIST, ITEM, ERR, ERROR,\*)
 

*Adds an item to the end of a double precision real list.*
- subroutine [LISTS::LIST\\_ITEM\\_IN\\_LIST\\_INTG1](#) (LIST, ITEM, LIST\_ITEM, ERR, ERROR,\*)
 

*Determines if ITEM is in the given integer LIST. If it is LIST\_ITEM is the index in the list. If not LIST\_ITEM is 0.*
- subroutine [LISTS::LIST\\_ITEM\\_IN\\_LIST\\_SP1](#) (LIST, ITEM, LIST\_ITEM, ERR, ERROR,\*)
 

*Determines if ITEM is in the given single precision real LIST. If it is LIST\_ITEM is the index in the list. If not LIST\_ITEM is 0.*
- subroutine [LISTS::LIST\\_ITEM\\_IN\\_LIST\\_DP1](#) (LIST, ITEM, LIST\_ITEM, ERR, ERROR,\*)
 

*Determines if ITEM is in the given double precision real LIST. If it is LIST\_ITEM is the index in the list. If not LIST\_ITEM is 0.*
- subroutine [LISTS::LIST\\_ITEM\\_DELETE](#) (LIST, LIST\_ITEM, ERR, ERROR,\*)
 

*Deletes the item given by the LIST\_ITEM index from the given list.*
- subroutine [LISTS::LIST\\_NUMBER\\_OF\\_ITEMS\\_GET](#) (LIST, NUMBER\_OF\_ITEMS, ERR, ERROR,\*)
 

*Gets the current number of items in a list.*

- subroutine [LISTS::LIST\\_DETACH\\_AND\\_DESTROY\\_INTG](#) (LIST, NUMBER\_IN\_LIST, LIST\_VALUES, ERR, ERROR,\*)

*Detaches the list values from an integer list and returns them as a pointer to a array of base type before destroying the list. The LIST\_VALUES pointer must not be associated on entry. It is up to the user to then deallocate the returned list memory.*

- subroutine [LISTS::LIST\\_DETACH\\_AND\\_DESTROY\\_SP](#) (LIST, NUMBER\_IN\_LIST, LIST\_VALUES, ERR, ERROR,\*)

*Detaches the list values from a single precision real list and returns them as a pointer to a array of base type before destroying the list. The LIST\_VALUES pointer must not be associated on entry. It is up to the user to then deallocate the returned list memory.*

- subroutine [LISTS::LIST\\_DETACH\\_AND\\_DESTROY\\_DP](#) (LIST, NUMBER\_IN\_LIST, LIST\_VALUES, ERR, ERROR,\*)

*Detaches the list values from a double precision real list and returns them as a pointer to a array of base type before destroying the list. The LIST\_VALUES pointer must not be associated on entry. It is up to the user to then deallocate the returned list memory.*

- subroutine [LISTS::LIST\\_REMOVE\\_DUPLICATES](#) (LIST, ERR, ERROR,\*)

*Removes duplicate entries from a list. A side effect of this is that the list is sorted.*

- subroutine [LISTS::LIST\\_SEARCH\\_INTG\\_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,\*)

*Searches an integer array list A for VALUE. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.*

- subroutine [LISTS::LIST\\_SEARCH\\_SP\\_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,\*)

*Searches a single precision real array list A for VALUE. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.*

- subroutine [LISTS::LIST\\_SEARCH\\_DP\\_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,\*)

*Searches a double precision real array list A for VALUE. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.*

- subroutine [LISTS::LIST\\_SEARCH\\_LINEAR\\_INTG\\_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,\*)

*Searches an integer array list A for VALUE using the linear search method. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.*

- subroutine [LISTS::LIST\\_SEARCH\\_LINEAR\\_SP\\_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,\*)

*Searches a single precision real array list A for VALUE using the linear search method. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.*

- subroutine [LISTS::LIST\\_SEARCH\\_LINEAR\\_DP\\_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,\*)

*Searches a double precision real array list A for VALUE using the linear search method. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.*

- subroutine [LISTS::LIST\\_SORT\\_INTG\\_ARRAY](#) (A, ERR, ERROR,\*)

*Sorts an integer array list into ascending order.*

- subroutine [LISTS::LIST\\_SORT\\_SP\\_ARRAY](#) (A, ERR, ERROR,\*)

*Sorts an single precision array list into ascending order.*

- subroutine [LISTS::LIST\\_SORT\\_DP\\_ARRAY](#) (A, ERR, ERROR,\*)
 

*Sorts an double precision array list into ascending order.*
- subroutine [LISTS::LIST\\_SORT\\_BUBBLE\\_INTG\\_ARRAY](#) (A, ERR, ERROR,\*)
 

*BUBBLE\_SORT\_INTG performs a bubble sort on an integer array list.*
- subroutine [LISTS::LIST\\_SORT\\_BUBBLE\\_SP\\_ARRAY](#) (A, ERR, ERROR,\*)
 

*BUBBLE\_SORT\_SP performs a bubble sort on a single precision array list.*
- subroutine [LISTS::LIST\\_SORT\\_BUBBLE\\_DP\\_ARRAY](#) (A, ERR, ERROR,\*)
 

*BUBBLE\_SORT\_DP performs a bubble sort on a double precision list.*
- subroutine [LISTS::LIST\\_SORT\\_HEAP\\_INTG\\_ARRAY](#) (A, ERR, ERROR,\*)
 

*Sorts an integer array list into assending order using the heap sort method.*
- subroutine [LISTS::LIST\\_SORT\\_HEAP\\_SP\\_ARRAY](#) (A, ERR, ERROR,\*)
 

*Sorts a real single precision array list into assending order using the heap sort method.*
- subroutine [LISTS::LIST\\_SORT\\_HEAP\\_DP\\_ARRAY](#) (A, ERR, ERROR,\*)
 

*Sorts a real double precision array list into assending order using the heap sort method.*
- subroutine [LISTS::LIST\\_SORT\\_SHELL\\_INTG\\_ARRAY](#) (A, ERR, ERROR,\*)
 

*Sorts an integer array list into either assending or descending order using the shell sort method.*
- subroutine [LISTS::LIST\\_SORT\\_SHELL\\_SP\\_ARRAY](#) (A, ERR, ERROR,\*)
 

*Sorts a real single precision array list into either assending or descending order using the shell sort method.*
- subroutine [LISTS::LIST\\_SORT\\_SHELL\\_DP\\_ARRAY](#) (A, ERR, ERROR,\*)
 

*Sorts a real double precision array list into either assending or descending order using the shell sort method.*

## Variables

- INTEGER(INTG), parameter [LISTS::LIST\\_INTG\\_TYPE](#) = INTEGER\_TYPE
 

*Integer data type for a list.*
- INTEGER(INTG), parameter [LISTS::LIST\\_SP\\_TYPE](#) = SINGLE\_REAL\_TYPE
 

*Single precision real data type for a list.*
- INTEGER(INTG), parameter [LISTS::LIST\\_DP\\_TYPE](#) = DOUBLE\_REAL\_TYPE
 

*Double precision real data type for a list.*
- INTEGER(INTG), parameter [LISTS::LIST\\_UNSORTED\\_TYPE](#) = 1
 

*Unsorted list type.*
- INTEGER(INTG), parameter [LISTS::LIST\\_SORT\\_ASCENDING\\_TYPE](#) = 2
 

*Ascending order for sort.*

- INTEGER(INTG), parameter **LISTS::LIST\_SORT\_DESCENDING\_TYPE** = 3  
*Descending order for sort.*
- INTEGER(INTG), parameter **LISTS::LIST\_BUBBLE\_SORT\_METHOD** = 1  
*Bubble sort method.*
- INTEGER(INTG), parameter **LISTS::LIST\_SHELL\_SORT\_METHOD** = 2  
*Shell sort method.*
- INTEGER(INTG), parameter **LISTS::LIST\_HEAP\_SORT\_METHOD** = 3  
*Heap sort method.*

### 8.40.1 Detailed Description

#### Id

[lists.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

#### Author:

Chris Bradley Implements lists of base types.

#### Todo

Fix up and have this module use the sorting module for sorts

### 8.40.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [lists.f90](#).

## **8.41 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine\_constants\_aix.f90 File Reference**

### **Id**

[machine\\_constants\\_aix.f90](#) 177 2008-10-25 13:38:18Z chrispb

### **Namespaces**

- namespace **MACHINE\_CONSTANTS**

*This module contains all machine dependent constants for AIX systems.*

### **Variables**

- INTEGER(INTG), parameter **MACHINE\_CONSTANTS::MACHINE\_TYPE** = IBM\_COMPUTER
- INTEGER(INTG), parameter **MACHINE\_CONSTANTS::MACHINE\_OS** = AIX\_OS
- INTEGER(INTG), parameter **MACHINE\_CONSTANTS::MACHINE\_ENDIAN** = LITTLE\_ENDIAN\_NUMBER
- INTEGER(INTG), parameter **MACHINE\_CONSTANTS::MACHINE\_CHAR\_FORMAT** = ASCII\_CHARACTER
- INTEGER(INTG), parameter **MACHINE\_CONSTANTS::MACHINE\_INT\_FORMAT** = TWOS\_COMPLEMENT\_INTEGER
- INTEGER(INTG), parameter **MACHINE\_CONSTANTS::MACHINE\_SP\_FORMAT** = SPIEEE\_NUMBER
- INTEGER(INTG), parameter **MACHINE\_CONSTANTS::MACHINE\_DP\_FORMAT** = DPIEEE\_NUMBER
- INTEGER(INTG), parameter **MACHINE\_CONSTANTS::INTEGER\_SIZE** = 4
- INTEGER(INTG), parameter **MACHINE\_CONSTANTS::SHORT\_INTEGER\_SIZE** = 2
- INTEGER(INTG), parameter **MACHINE\_CONSTANTS::LONG\_INTEGER\_SIZE** = 8
- INTEGER(INTG), parameter **MACHINE\_CONSTANTS::SINGLE\_REAL\_SIZE** = 4
- INTEGER(INTG), parameter **MACHINE\_CONSTANTS::DOUBLE\_REAL\_SIZE** = 8
- INTEGER(INTG), parameter **MACHINE\_CONSTANTS::CHARACTER\_SIZE** = 1
- INTEGER(INTG), parameter **MACHINE\_CONSTANTS::LOGICAL\_SIZE** = 4
- INTEGER(INTG), parameter **MACHINE\_CONSTANTS::SINGLE\_COMPLEX\_SIZE** = 8
- INTEGER(INTG), parameter **MACHINE\_CONSTANTS::DOUBLE\_COMPLEX\_SIZE** = 16
- CHARACTER(LEN=1), parameter **MACHINE\_CONSTANTS::ERROR\_SEPARATOR\_CONSTANT** = CHAR(6)

### **8.41.1 Detailed Description**

#### **Id**

[machine\\_constants\\_aix.f90](#) 177 2008-10-25 13:38:18Z chrispb

#### **Author:**

Chris Bradley This module contains all machine dependent constants for AIX systems.

### 8.41.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [machine\\_constants\\_aix.f90](#).

## **8.42 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine\_constants\_irix.f90 File Reference**

### **Id**

[machine\\_constants\\_irix.f90](#) 177 2008-10-25 13:38:18Z chrispb

### **Namespaces**

- namespace [MACHINE\\_CONSTANTS](#)

*This module contains all machine dependent constants for AIX systems.*

#### **8.42.1 Detailed Description**

##### **Id**

[machine\\_constants\\_irix.f90](#) 177 2008-10-25 13:38:18Z chrispb

##### **Author:**

Chris Bradley This module contains all machine dependent constants for IRIX systems.

#### **8.42.2 LICENSE**

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [machine\\_constants\\_irix.f90](#).

## 8.43 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine\_- constants\_linux.f90 File Reference

### Id

[machine\\_constants\\_linux.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

### Namespaces

- namespace [MACHINE\\_CONSTANTS](#)

*This module contains all machine dependent constants for AIX systems.*

### 8.43.1 Detailed Description

#### Id

[machine\\_constants\\_linux.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

#### Author:

Chris Bradley This module contains all machine dependent constants for Linux systems.

### 8.43.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [machine\\_constants\\_linux.f90](#).

## **8.44 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine\_constants\_vms.f90 File Reference**

### **Id**

[machine\\_constants\\_vms.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

### **Namespaces**

- namespace [MACHINE\\_CONSTANTS](#)

*This module contains all machine dependent constants for AIX systems.*

### **8.44.1 Detailed Description**

#### **Id**

[machine\\_constants\\_vms.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

#### **Author:**

Chris Bradley This module contains all machine dependent constants for VMS systems.

### **8.44.2 LICENSE**

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [machine\\_constants\\_vms.f90](#).

## 8.45 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine\_- constants\_win32.f90 File Reference

### Id

[machine\\_constants\\_win32.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

### Namespaces

- namespace [MACHINE\\_CONSTANTS](#)

*This module contains all machine dependent constants for AIX systems.*

### 8.45.1 Detailed Description

#### Id

[machine\\_constants\\_win32.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

#### Author:

Chris Bradley This module contains all machine dependent constants for Win32 systems.

### 8.45.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [machine\\_constants\\_win32.f90](#).

## 8.46 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/math.f90 File Reference

### Id

[math.f90](#) 181 2008-10-26 18:59:43Z chrisbradley

### Classes

- interface [MATHS::CROSS\\_PRODUCT](#)
- interface [MATHS::D\\_CROSS\\_PRODUCT](#)
- interface [MATHS::DETERMINANT](#)
- interface [MATHS::EDP](#)
- interface [MATHS::EIGENVALUE](#)
- interface [MATHS::EIGENVECTOR](#)
- interface [MATHS::IO](#)
- interface [MATHS::I1](#)
- interface [MATHS::INVERT](#)
- interface [MATHS::K0](#)
- interface [MATHS::K1](#)
- interface [MATHS::L2NORM](#)
- interface [MATHS::MATRIX\\_PRODUCT](#)
- interface [MATHS::MATRIX\\_TRANSPOSE](#)
- interface [MATHS::NORMALISE](#)
- interface [MATHS::SOLVE\\_SMALL\\_LINEAR\\_SYSTEM](#)

### Namespaces

- namespace [MATHS](#)

*This module contains all mathematics support routines.*

### Functions

- subroutine [MATHS::CROSS\\_PRODUCT\\_INTG](#) (A, B, C, ERR, ERROR,\*)
- subroutine [MATHS::CROSS\\_PRODUCT\\_SP](#) (A, B, C, ERR, ERROR,\*)
- subroutine [MATHS::CROSS\\_PRODUCT\\_DP](#) (A, B, C, ERR, ERROR,\*)
- subroutine [MATHS::D\\_CROSS\\_PRODUCT\\_INTG](#) (N, A, B, C, D\_A, D\_B, D\_C, ERR, ERROR,\*)
- subroutine [MATHS::D\\_CROSS\\_PRODUCT\\_SP](#) (N, A, B, C, D\_A, D\_B, D\_C, ERR, ERROR,\*)
- subroutine [MATHS::D\\_CROSS\\_PRODUCT\\_DP](#) (N, A, B, C, D\_A, D\_B, D\_C, ERR, ERROR,\*)
- INTEGER(INTG) [MATHS::DETERMINANT\\_FULL\\_INTG](#) (A, ERR, ERROR)
- REAL(SP) [MATHS::DETERMINANT\\_FULL\\_SP](#) (A, ERR, ERROR)
- REAL(DP) [MATHS::DETERMINANT\\_FULL\\_DP](#) (A, ERR, ERROR)
- REAL(DP) [MATHS::EDP\\_DP](#) (X)
- REAL(SP) [MATHS::EDP\\_SP](#) (X)
- subroutine [MATHS::EIGENVALUE\\_FULL\\_SP](#) (A, EVALUES, ERR, ERROR,\*)
- subroutine [MATHS::EIGENVALUE\\_FULL\\_DP](#) (A, EVALUES, ERR, ERROR,\*)
- subroutine [MATHS::EIGENVECTOR\\_FULL\\_SP](#) (A, EVALUE, EVECTOR, ERR, ERROR,\*)
- subroutine [MATHS::EIGENVECTOR\\_FULL\\_DP](#) (A, EVALUE, EVECTOR, ERR, ERROR,\*)

- REAL(DP) [MATHS::I0\\_DP](#) (X)
- REAL(SP) [MATHS::I0\\_SP](#) (X)
- REAL(DP) [MATHS::I1\\_DP](#) (X)
- REAL(SP) [MATHS::I1\\_SP](#) (X)
- subroutine [MATHS::INVERT\\_FULL\\_SP](#) (A, B, DET, ERR, ERROR,\*)
- subroutine [MATHS::INVERT\\_FULL\\_DP](#) (A, B, DET, ERR, ERROR,\*)
- REAL(DP) [MATHS::K0\\_DP](#) (X)
- REAL(SP) [MATHS::K0\\_SP](#) (X)
- REAL(DP) [MATHS::K1\\_DP](#) (X)
- REAL(SP) [MATHS::K1\\_SP](#) (X)
- REAL(DP) [MATHS::KDP\\_DP](#) (X)
- REAL(SP) [MATHS::KDP\\_SP](#) (X)
- REAL(SP) [MATHS::L2NORM\\_SP](#) (A)
- REAL(DP) [MATHS::L2NORM\\_DP](#) (A)
- subroutine [MATHS::MATRIX\\_PRODUCT\\_SP](#) (A, B, C, ERR, ERROR,\*)
- subroutine [MATHS::MATRIX\\_PRODUCT\\_DP](#) (A, B, C, ERR, ERROR,\*)
- subroutine [MATHS::MATRIX\\_TRANSPOSE\\_SP](#) (A, AT, ERR, ERROR,\*)
- subroutine [MATHS::MATRIX\\_TRANSPOSE\\_DP](#) (A, AT, ERR, ERROR,\*)
- REAL(SP) [MATHS::NORMALISE\\_SP](#) (A, ERR, ERROR)
- REAL(DP) [MATHS::NORMALISE\\_DP](#) (A, ERR, ERROR)
- subroutine [MATHS::SOLVE\\_SMALL\\_LINEAR\\_SYSTEM\\_SP](#) (A, x, b, ERR, ERROR,\*)
- subroutine [MATHS::SOLVE\\_SMALL\\_LINEAR\\_SYSTEM\\_DP](#) (A, x, b, ERR, ERROR,\*)

### 8.46.1 Detailed Description

#### Id

[maths.f90](#) 181 2008-10-26 18:59:43Z chrisbradley

#### Author:

Chris Bradley This module contains all mathematics support routines.

### 8.46.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s): Kumar Mithraratne

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [math.f90](#).

## 8.47 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/matrix\_vector.f90 File Reference

### Id

[matrix\\_vector.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

### Classes

- interface [MATRIX\\_VECTOR::MATRIX\\_ALL\\_VALUES\\_SET](#)
- interface [MATRIX\\_VECTOR::MATRIX\\_DATA\\_GET](#)
- interface [MATRIX\\_VECTOR::MATRIX\\_VALUES\\_ADD](#)
- interface [MATRIX\\_VECTOR::MATRIX\\_VALUES\\_GET](#)
- interface [MATRIX\\_VECTOR::MATRIX\\_VALUES\\_SET](#)
- interface [MATRIX\\_VECTOR::VECTOR\\_ALL\\_VALUES\\_SET](#)
- interface [MATRIX\\_VECTOR::VECTOR\\_DATA\\_GET](#)
- interface [MATRIX\\_VECTOR::VECTOR\\_VALUES\\_GET](#)
- interface [MATRIX\\_VECTOR::VECTOR\\_VALUES\\_SET](#)

### Namespaces

- namespace [MATRIX\\_VECTOR](#)

*This module contains all routines dealing with (non-distributed) matrix and vectors types.*

### Functions

- subroutine [MATRIX\\_VECTOR::MATRIX\\_ALL\\_VALUES\\_SET\\_INTG](#) (MATRIX, VALUE, ERR, ERROR,\*)
 

*Sets all values in an integer matrix to the specified value.*
- subroutine [MATRIX\\_VECTOR::MATRIX\\_ALL\\_VALUES\\_SET\\_SP](#) (MATRIX, VALUE, ERR, ERROR,\*)
 

*Sets all values in a single precision matrix to the specified value.*
- subroutine [MATRIX\\_VECTOR::MATRIX\\_ALL\\_VALUES\\_SET\\_DP](#) (MATRIX, VALUE, ERR, ERROR,\*)
 

*Sets all values in a double precision matrix to the specified value.*
- subroutine [MATRIX\\_VECTOR::MATRIX\\_ALL\\_VALUES\\_SET\\_L](#) (MATRIX, VALUE, ERR, ERROR,\*)
 

*Sets all values in a logical matrix to the specified value.*
- subroutine [MATRIX\\_VECTOR::MATRIX\\_CREATE\\_FINISH](#) (MATRIX, ERR, ERROR,\*)
 

*Finishes the creation a matrix.*
- subroutine [MATRIX\\_VECTOR::MATRIX\\_CREATE\\_START](#) (MATRIX, ERR, ERROR,\*)
 

*Starts the creation a matrix.*

- subroutine `MATRIX_VECTOR::MATRIX_DATA_GET_INTG` (MATRIX, DATA, ERR, ERROR,\*)

*Returns a pointer to the data of an integer matrix. Note: the values can be used for read operations but a `MATRIX_VALUES_SET` call must be used to change any values. The pointer should not be deallocated.*

- subroutine `MATRIX_VECTOR::MATRIX_DATA_GET_SP` (MATRIX, DATA, ERR, ERROR,\*)

*Returns a pointer to the data of a single precision matrix. Note: the values can be used for read operations but a `MATRIX_VALUES_SET` call must be used to change any values. The pointer should not be deallocated.*

- subroutine `MATRIX_VECTOR::MATRIX_DATA_GET_DP` (MATRIX, DATA, ERR, ERROR,\*)

*Returns a pointer to the data of a double precision matrix. Note: the values can be used for read operations but a `MATRIX_VALUES_SET` call must be used to change any values. The pointer should not be deallocated.*

- subroutine `MATRIX_VECTOR::MATRIX_DATA_GET_L` (MATRIX, DATA, ERR, ERROR,\*)

*Returns a pointer to the data of a logical matrix. Note: the values can be used for read operations but a `MATRIX_VALUES_SET` call must be used to change any values. The pointer should not be deallocated.*

- subroutine `MATRIX_VECTOR::MATRIX_DATA_TYPE_SET` (MATRIX, DATA\_TYPE, ERR, ERROR,\*)

*Sets/changes the data type of a matrix.*

- subroutine `MATRIX_VECTOR::MATRIX_DESTROY` (MATRIX, ERR, ERROR,\*)

*Destroys a matrix.*

- subroutine `MATRIX_VECTOR::MATRIX_DUPLICATE` (MATRIX, NEW\_MATRIX, ERR, ERROR,\*)

*Duplicates the matrix and returns a pointer to the duplicated matrix in NEWMATRIX.*

- subroutine `MATRIX_VECTOR::MATRIX_FINALISE` (MATRIX, ERR, ERROR,\*)

*Finalises a matrix and deallocates all memory.*

- subroutine `MATRIX_VECTOR::MATRIX_INITIALISE` (MATRIX, ERR, ERROR,\*)

*Initialises a matrix.*

- subroutine `MATRIX_VECTOR::MATRIX_MAX_COLUMNS_PER_ROW_GET` (MATRIX, MAX\_COLUMNS\_PER\_ROW, ERR, ERROR,\*)

*Gets the maximum number of columns in each row of a distributed matrix.*

- subroutine `MATRIX_VECTOR::MATRIX_NUMBER_NON_ZEROS_SET` (MATRIX, NUMBER\_NON\_ZEROS, ERR, ERROR,\*)

*Sets/changes the number of non zeros for a matrix.*

- subroutine `MATRIX_VECTOR::MATRIX_MAX_SIZE_SET` (MATRIX, MAX\_M, MAX\_N, ERR, ERROR,\*)

*Sets/changes the maximum size of a matrix.*

- subroutine `MATRIX_VECTOR::MATRIX_OUTPUT` (ID, MATRIX, ERR, ERROR,\*)

*Sets/changes the size of a matrix.*

- subroutine `MATRIX_VECTOR::MATRIX_SIZE_SET` (MATRIX, M, N, ERR, ERROR,\*)

*Sets/changes the size of a matrix.*

- subroutine [MATRIX\\_VECTOR::MATRIX\\_STORAGE\\_LOCATION\\_FIND](#) (MATRIX, I, J, LOCATION, ERR, ERROR,\*)

*Returns the storage location in the data array of a matrix that corresponds to location I,J. If the location does not exist the routine returns zero.*

- subroutine [MATRIX\\_VECTOR::MATRIX\\_STORAGE\\_LOCATIONS\\_GET](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, ERR, ERROR,\*)

*Gets the storage locations (sparsity pattern) of a matrix.*

- subroutine [MATRIX\\_VECTOR::MATRIX\\_STORAGE\\_LOCATIONS\\_SET](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, ERR, ERROR,\*)

*Sets the storage locations (sparsity pattern) in a matrix to that specified by the row and column indices.*

- subroutine [MATRIX\\_VECTOR::MATRIX\\_STORAGE\\_TYPE\\_GET](#) (MATRIX, STORAGE\_TYPE, ERR, ERROR,\*)

*Gets the storage type for a matrix.*

- subroutine [MATRIX\\_VECTOR::MATRIX\\_STORAGE\\_TYPE\\_SET](#) (MATRIX, STORAGE\_TYPE, ERR, ERROR,\*)

*Sets/changes the storage type for a matrix.*

- subroutine [MATRIX\\_VECTOR::MATRIX\\_VALUES\\_ADD\\_INTG](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)

*Adds values to an integer matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.*

- subroutine [MATRIX\\_VECTOR::MATRIX\\_VALUES\\_ADD\\_INTG1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)

*Adds a value to an integer matrix at the location specified by the row and column index i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.*

- subroutine [MATRIX\\_VECTOR::MATRIX\\_VALUES\\_ADD\\_INTG2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)

*Adds a matrix of values to an integer matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.*

- subroutine [MATRIX\\_VECTOR::MATRIX\\_VALUES\\_ADD\\_SP](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)

*Adds values to a single precision real matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.*

- subroutine [MATRIX\\_VECTOR::MATRIX\\_VALUES\\_ADD\\_SP1](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)

*Adds a value to a single precision real matrix at the location specified by the row and column index i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.*

- subroutine [MATRIX\\_VECTOR::MATRIX\\_VALUES\\_ADD\\_SP2](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)

*Adds a matrix of values to a single precision real matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.*

- subroutine **MATRIX\_VECTOR::MATRIX\_VALUES\_ADD\_DP** (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Adds values to a double precision real matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I,J)=\text{MATRIX}(I,J)+\text{VALUE}$ .*
- subroutine **MATRIX\_VECTOR::MATRIX\_VALUES\_ADD\_DP1** (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
 

*Adds a value to a double precision real matrix at the location specified by the row and column index i.e.,  $\text{MATRIX}(I,J)=\text{MATRIX}(I,J)+\text{VALUE}$ .*
- subroutine **MATRIX\_VECTOR::MATRIX\_VALUES\_ADD\_DP2** (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Adds a matrix of values to a double precision real matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I,J)=\text{MATRIX}(I,J)+\text{VALUE}$ .*
- subroutine **MATRIX\_VECTOR::MATRIX\_VALUES\_ADD\_L** (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Adds values to a logical matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I,J)=\text{MATRIX}(I,J).\text{OR}.\text{VALUE}$ .*
- subroutine **MATRIX\_VECTOR::MATRIX\_VALUES\_ADD\_L1** (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
 

*Adds a value to a logical matrix at the location specified by the row and column index i.e.,  $\text{MATRIX}(I,J)=\text{MATRIX}(I,J).\text{OR}.\text{VALUE}$ .*
- subroutine **MATRIX\_VECTOR::MATRIX\_VALUES\_ADD\_L2** (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Adds a matrix of values to a logical matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I,J)=\text{MATRIX}(I,J).\text{OR}.\text{VALUE}$ .*
- subroutine **MATRIX\_VECTOR::MATRIX\_VALUES\_GET\_INTG** (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Gets the values in an integer matrix at the location specified by the row and column indices i.e.,  $\text{VALUE}=\text{MATRIX}(I,J)$ .*
- subroutine **MATRIX\_VECTOR::MATRIX\_VALUES\_GET\_INTG1** (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
 

*Gets a value in an integer matrix at the location specified by the row and column index i.e.,  $\text{VALUE}=\text{MATRIX}(I,J)$ .*
- subroutine **MATRIX\_VECTOR::MATRIX\_VALUES\_GET\_INTG2** (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Gets the matrix of values in an integer matrix at the location specified by the row and column indices i.e.,  $\text{VALUE}=\text{MATRIX}(I,J)$ .*
- subroutine **MATRIX\_VECTOR::MATRIX\_VALUES\_GET\_SP** (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Gets values in a single precision real matrix at the location specified by the row and column indices i.e.,  $\text{VALUE}=\text{MATRIX}(I,J)$ .*

- subroutine [\*\*MATRIX\\_VECTOR::MATRIX\\_VALUES\\_GET\\_SP1\*\*](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
 

*Gets a value in a single precision real matrix at the location specified by the row and column index i.e., VALUE=MATRIX(I,J).*
- subroutine [\*\*MATRIX\\_VECTOR::MATRIX\\_VALUES\\_GET\\_SP2\*\*](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Gets a matrix of values in a single precision real matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).*
- subroutine [\*\*MATRIX\\_VECTOR::MATRIX\\_VALUES\\_GET\\_DP\*\*](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Gets values in a double precision real matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).*
- subroutine [\*\*MATRIX\\_VECTOR::MATRIX\\_VALUES\\_GET\\_DP1\*\*](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
 

*Gets a value in a double precision real matrix at the location specified by the row and column index i.e., VALUE=MATRIX(I,J).*
- subroutine [\*\*MATRIX\\_VECTOR::MATRIX\\_VALUES\\_GET\\_DP2\*\*](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Gets a matrix of values in a double precision real matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).*
- subroutine [\*\*MATRIX\\_VECTOR::MATRIX\\_VALUES\\_GET\\_L\*\*](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Gets values in a logical matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).*
- subroutine [\*\*MATRIX\\_VECTOR::MATRIX\\_VALUES\\_GET\\_L1\*\*](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
 

*Gets a value in a logical matrix at the location specified by the row and column index i.e., VALUE=MATRIX(I,J).*
- subroutine [\*\*MATRIX\\_VECTOR::MATRIX\\_VALUES\\_GET\\_L2\*\*](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Gets a matrix of values in a logical matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).*
- subroutine [\*\*MATRIX\\_VECTOR::MATRIX\\_VALUES\\_SET\\_INTG\*\*](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Sets the values in an integer matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=VALUE.*
- subroutine [\*\*MATRIX\\_VECTOR::MATRIX\\_VALUES\\_SET\\_INTG1\*\*](#) (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)
 

*Sets a value in an integer matrix at the location specified by the row and column index i.e., MATRIX(I,J)=VALUE.*
- subroutine [\*\*MATRIX\\_VECTOR::MATRIX\\_VALUES\\_SET\\_INTG2\*\*](#) (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)
 

*Sets a matrix of values in an integer matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=VALUE.*

*Sets the matrix of values in an integer matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I,J)=\text{VALUE}$ .*

- subroutine **MATRIX\_VECTOR::MATRIX\_VALUES\_SET\_SP** (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)

*Sets the values in a single precision real matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I,J)=\text{VALUE}$ .*

- subroutine **MATRIX\_VECTOR::MATRIX\_VALUES\_SET\_SP1** (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)

*Sets the value in a single precision real matrix at the location specified by the row and column index i.e.,  $\text{MATRIX}(I,J)=\text{VALUE}$ .*

- subroutine **MATRIX\_VECTOR::MATRIX\_VALUES\_SET\_SP2** (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)

*Sets the matrix of values in a single precision real matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I,J)=\text{VALUE}$ .*

- subroutine **MATRIX\_VECTOR::MATRIX\_VALUES\_SET\_DP** (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)

*Sets the values in a double precision real matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I,J)=\text{VALUE}$ .*

- subroutine **MATRIX\_VECTOR::MATRIX\_VALUES\_SET\_DP1** (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)

*Sets a value in a double precision real matrix at the location specified by the row and column index i.e.,  $\text{MATRIX}(I,J)=\text{VALUE}$ .*

- subroutine **MATRIX\_VECTOR::MATRIX\_VALUES\_SET\_DP2** (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)

*Sets the matrix of values in a double precision real matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I,J)=\text{VALUE}$ .*

- subroutine **MATRIX\_VECTOR::MATRIX\_VALUES\_SET\_L** (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)

*Sets the values in a logical matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I,J)=\text{VALUE}$ .*

- subroutine **MATRIX\_VECTOR::MATRIX\_VALUES\_SET\_L1** (MATRIX, ROW\_INDEX, COLUMN\_INDEX, VALUE, ERR, ERROR,\*)

*Sets a value in a logical matrix at the location specified by the row and column index i.e.,  $\text{MATRIX}(I,J)=\text{VALUE}$ .*

- subroutine **MATRIX\_VECTOR::MATRIX\_VALUES\_SET\_L2** (MATRIX, ROW\_INDICES, COLUMN\_INDICES, VALUES, ERR, ERROR,\*)

*Sets the matrix of values in a logical matrix at the location specified by the row and column indices i.e.,  $\text{MATRIX}(I,J)=\text{VALUE}$ .*

- subroutine **MATRIX\_VECTOR::VECTOR\_ALL\_VALUES\_SET\_INTG** (VECTOR, VALUE, ERR, ERROR,\*)

*Sets all values in an integer vector to the specified value.*

- subroutine [MATRIX\\_VECTOR::VECTOR\\_ALL\\_VALUES\\_SET\\_SP](#) (VECTOR, VALUE, ERR, ERROR,\*)
 

*Sets all values in a single precision vector to the specified value.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_ALL\\_VALUES\\_SET\\_DP](#) (VECTOR, VALUE, ERR, ERROR,\*)
 

*Sets all values in a double precision vector to the specified value.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_ALL\\_VALUES\\_SET\\_L](#) (VECTOR, VALUE, ERR, ERROR,\*)
 

*Sets all values in a logical vector to the specified value.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_CREATE\\_FINISH](#) (VECTOR, ERR, ERROR,\*)
 

*Finishes the creation of a vector.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_CREATE\\_START](#) (VECTOR, ERR, ERROR,\*)
 

*Starts the creation a vector.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_DATA\\_GET\\_INTG](#) (VECTOR, DATA, ERR, ERROR,\*)
 

*Returns a pointer to the data of an integer vector. Note: the values can be used for read operations but a [VECTOR\\_VALUES\\_SET](#) call must be used to change any values. The pointer should not be deallocated.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_DATA\\_GET\\_SP](#) (VECTOR, DATA, ERR, ERROR,\*)
 

*Returns a pointer to the data of a single precision vector. Note: the values can be used for read operations but a [VECTOR\\_VALUES\\_SET](#) call must be used to change any values. The pointer should not be deallocated.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_DATA\\_GET\\_DP](#) (VECTOR, DATA, ERR, ERROR,\*)
 

*Returns a pointer to the data of a double precision vector. Note: the values can be used for read operations but a [VECTOR\\_VALUES\\_SET](#) call must be used to change any values. The pointer should not be deallocated.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_DATA\\_GET\\_L](#) (VECTOR, DATA, ERR, ERROR,\*)
 

*Returns a pointer to the data of a logical vector. Note: the values can be used for read operations but a [VECTOR\\_VALUES\\_SET](#) call must be used to change any values. The pointer should not be deallocated.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_DATA\\_TYPE\\_SET](#) (VECTOR, DATA\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the data type of a vector.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_DESTROY](#) (VECTOR, ERR, ERROR,\*)
 

*Destroys a vector.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_DUPLICATE](#) (VECTOR, NEW\_VECTOR, ERR, ERROR,\*)
 

*Duplicates a vector structure and returns a pointer to the new vector in NEW\_VECTOR.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_FINALISE](#) (VECTOR, ERR, ERROR,\*)
 

*Finalises a vector and deallocates all memory.*

- subroutine [MATRIX\\_VECTOR::VECTOR\\_INITIALISE](#) (VECTOR, ERR, ERROR,\*)
 

*Initialises a vector.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_SIZE\\_SET](#) (VECTOR, N, ERR, ERROR,\*)
 

*Sets/changes the size of a vector.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_VALUES\\_GET\\_INTG](#) (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
 

*Gets the values in an integer vector at the indices specified.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_VALUES\\_GET\\_INTG1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
 

*Gets a value in an integer vector at the location specified by the index.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_VALUES\\_GET\\_SP](#) (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
 

*Gets the values in a single precision real vector at the indices specified.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_VALUES\\_GET\\_SP1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
 

*Gets a value in a single precision vector at the location specified by the index.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_VALUES\\_GET\\_DP](#) (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
 

*Gets the values in a double precision real vector at the indices specified.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_VALUES\\_GET\\_DP1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
 

*Gets a value in a double precision vector at the location specified by the index.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_VALUES\\_GET\\_L](#) (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
 

*Gets the values in a logical real vector at the indices specified.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_VALUES\\_GET\\_L1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
 

*Gets a value in a logical vector at the location specified by the index.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_VALUES\\_SET\\_INTG](#) (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
 

*Sets the values in an integer vector at the specified indices.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_VALUES\\_SET\\_INTG1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
 

*Sets a value in an integer vector at the specified index.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_VALUES\\_SET\\_SP](#) (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
 

*Sets the values in a single precision vector at the specified indices.*

- subroutine [MATRIX\\_VECTOR::VECTOR\\_VALUES\\_SET\\_SP1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
 

*Sets a value in a single precision vector at the specified index.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_VALUES\\_SET\\_DP](#) (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
 

*Sets the values in a double precision vector at the specified indices.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_VALUES\\_SET\\_DPI](#) (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
 

*Sets a value in a double precision vector at the specified index.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_VALUES\\_SET\\_L](#) (VECTOR, INDICES, VALUES, ERR, ERROR,\*)
 

*Sets the values in a logical vector at the specified indices.*
- subroutine [MATRIX\\_VECTOR::VECTOR\\_VALUES\\_SET\\_L1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,\*)
 

*Sets a value in a logical vector at the specified index.*

## Variables

- INTEGER(INTG), parameter [MATRIX\\_VECTOR::MATRIX\\_VECTOR\\_INTG\\_TYPE](#) = INTEGER\_TYPE
 

*Integer matrix-vector data type.*
- INTEGER(INTG), parameter [MATRIX\\_VECTOR::MATRIX\\_VECTOR\\_SP\\_TYPE](#) = SINGLE\_REAL\_TYPE
 

*Single precision real matrix-vector data type.*
- INTEGER(INTG), parameter [MATRIX\\_VECTOR::MATRIX\\_VECTOR\\_DP\\_TYPE](#) = DOUBLE\_REAL\_TYPE
 

*Double precision real matrix-vector data type.*
- INTEGER(INTG), parameter [MATRIX\\_VECTOR::MATRIX\\_VECTOR\\_L\\_TYPE](#) = LOGICAL\_TYPE
 

*Logical matrix-vector data type.*
- INTEGER(INTG), parameter [MATRIX\\_VECTOR::MATRIX\\_BLOCK\\_STORAGE\\_TYPE](#) = 0
 

*Matrix block storage type.*
- INTEGER(INTG), parameter [MATRIX\\_VECTOR::MATRIX\\_DIAGONAL\\_STORAGE\\_TYPE](#) = 1
 

*Matrix diagonal storage type.*
- INTEGER(INTG), parameter [MATRIX\\_VECTOR::MATRIX\\_COLUMN\\_MAJOR\\_STORAGE\\_TYPE](#) = 2
 

*Matrix column major storage type.*

- INTEGER(INTG), parameter **MATRIX\_VECTOR::MATRIX\_ROW\_MAJOR\_STORAGE\_TYPE = 3**  
*Matrix row major storage type.*
- INTEGER(INTG), parameter **MATRIX\_VECTOR::MATRIX\_COMPRESSED\_ROW\_STORAGE\_TYPE = 4**  
*Matrix compressed row storage type.*
- INTEGER(INTG), parameter **MATRIX\_VECTOR::MATRIX\_COMPRESSED\_COLUMN\_STORAGE\_TYPE = 5**  
*Matrix compressed column storage type.*
- INTEGER(INTG), parameter **MATRIX\_VECTOR::MATRIX\_ROW\_COLUMN\_STORAGE\_TYPE = 6**  
*Matrix row-column storage type.*
- INTEGER(INTG), save **MATRIX\_VECTOR::MATRIX\_VECTOR\_ID = 1**

### 8.47.1 Detailed Description

#### Id

[matrix\\_vector.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

#### Author:

Chris Bradley This module contains all routines dealing with (non-distributed) matrix and vectors types.

### 8.47.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

#### Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL

or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

### 8.47.3 MATRIX STORAGE STRUCTURES

The matrix storage structures used are governed by the STORAGE parameter associated with the array. If STORAGE is MATRIX\_BLOCK\_STORAGE\_TYPE the matrix is not sparse and the non-sparse matrix dimension M is used to calculate the matrix storage locations. If storage is MATRIX\_DIAGONAL\_STORAGE\_TYPE then only the matrix diagonal is stored. If storage is MATRIX\_COLUMN\_MAJOR\_STORAGE\_TYPE the matrix is not sparse and the non-sparse matrix dimension MAX\_M ( $>=M$ ) is used to calculate the matrix storage locations. If storage is MATRIX\_ROW\_MAJOR\_STORAGE\_TYPE the matrix is not sparse and the non-sparse matrix dimension MAX\_N ( $>=N$ ) is used to calculate the matrix storage locations. If STORAGE is MATRIX\_COMPRESSED\_ROW\_STORAGE\_TYPE the matrix has compressed row storage/sparsity (see below) and the sparsity structure arrays ROW\_INDICES and COLUMN\_INDICES are used for the storage location calculation. If STORAGE is MATRIX\_COMPRESSED\_COLUMN\_STORAGE\_TYPE the matrix has compressed column storage/sparsity (see below) and the sparsity structure arrays ROW\_INDICES and COLUMN\_INDICES are used for the storage location calculation. If STORAGE is MATRIX\_ROW\_COLUMN\_STORAGE\_TYPE the matrix has row column storage/sparsity (see below) and the sparsity structure arrays ROW\_INDICES and COLUMN\_INDICES are used for the storage location calculation.

#### 8.47.3.1 COMPRESSED-ROW STORAGE:

The storage structure scheme is based on storing a  $M \times N$  matrix as a one dimensional array of length SIZE (=NUMBER\_NON\_ZEROS) (where NUMBER\_NON\_ZEROS= $s \times M \times N$ , s is the sparsity of the array) that stores only the non-zero elements of the matrix. Two additional arrays ROW\_INDICES and COLUMN\_INDICES store the positions of the non-zero elements. ROW\_INDICES is of length  $M+1$  and COLUMN is of length NUMBER\_NON\_ZEROS. ROW\_INDICES(i) stores the position in COLUMN\_INDICES of the start of row i. The  $M+1$  position of ROW\_INDICES stores the size of COLUMN\_INDICES+1 i.e., NUMBER\_NON\_ZEROS+1. The number of non-zero elements in row i can be found from ROW\_INDICES(i+1)-ROW\_INDICES(i). COLUMN\_INDICES(nz) gives the column number for non-zero element nz. See also COMPRESSED-COLUMN storage.

Example of the compressed-row storage scheme on a  $N \times N$  matrix ( $N=6$ ). Here the sparsity is 8/36 or 22%

| GX | 1 | 2 | 3 | 4 | 5 | 6 |  |
|----|---|---|---|---|---|---|--|
| 1  | 0 | A | 0 | B | 0 | 0 |  |
| 2  | 0 | 0 | C | 0 | 0 | 0 |  |
| 3  | 0 | 0 | 0 | 0 | D | E |  |
| 4  | F | 0 | 0 | 0 | 0 | 0 |  |
| 5  | 0 | 0 | G | 0 | 0 | 0 |  |
| 6  | 0 | 0 | 0 | 0 | 0 | H |  |

  

| DATA (nz)          |   |   |   |   |   |   |   |
|--------------------|---|---|---|---|---|---|---|
| A                  | B | C | D | E | F | G | H |
|                    |   |   |   |   |   |   |   |
| ROW_INDICES (i)    |   |   |   |   |   |   |   |
| 1                  | 3 | 4 | 6 | 7 | 8 | 9 |   |
| COLUMN_INDICES (i) |   |   |   |   |   |   |   |
| 2                  | 4 | 3 | 5 | 6 | 1 | 3 | 6 |

#### 8.47.3.2 COMPRESSED-COLUMN STORAGE:

The storage structure scheme is based on storing a  $M \times N$  matrix as a one dimensional array of length SIZE (=NUMBER\_NON\_ZEROS) (where NUMBER\_NON\_ZEROS= $s \times M \times N$ , s is the sparsity of the array) that stores only the non-zero elements of the matrix. Two additional arrays ROW\_INDICES and COLUMN\_INDICES store the positions of the non-zero elements. ROW\_INDICES is of length NUMBER\_NON\_ZEROS and COLUMN is of length  $N+1$ . COLUMN\_INDICES(j) stores the position in ROW\_INDICES

of the start of column j. The N+1 position of COLUMN\_INDICES stores the size of ROW\_INDICES+1 i.e., NUMBER\_NON\_ZEROS+1. The number of non-zero elements in column j can be found from COLUMN\_INDICES(j+1)-COLUMN\_INDICES(j). ROW\_INDICES(nz) gives the row number for non-zero element nz. See also COMPRESSED-ROW storage.

Example of compressed-column storage scheme on a NxN matrix (N=6). Here the sparsity is 8/36 or 22%

|                 |                    |
|-----------------|--------------------|
| GX 1 2 3 4 5 6  |                    |
| <hr/>           |                    |
| 1   0 A 0 B 0 0 | DATA (nz)          |
| 2   0 0 C 0 0 0 | F A C G B D E H    |
| 3   0 0 0 0 D E | ROW_INDICES (i)    |
| 4   F 0 0 0 0 0 | 4 1 2 5 1 3 3 6    |
| 5   0 0 G 0 0 0 | COLUMN_INDICES (i) |
| 6   0 0 0 0 0 H | 1 2 3 5 6 7 9      |

#### 8.47.3.3 ROW-COLUMN STORAGE:

The storage structure scheme is based on storing a MxN matrix as a one dimensional array of length SIZE (=NUMBER\_NON\_ZEROS) (where NUMBER\_NON\_ZEROS=sxMxN, s is the sparsity of the array) that stores only the non-zero elements of the matrix. Two additional arrays ROW\_INDICES and COLUMN\_INDICES store the positions of the non-zero elements. Both ROW\_INDICES and COLUMN\_INDICES are of length NUMBER\_NON\_ZEROS. ROW\_INDICES(nz) gives the row number for non-zero element nz and COLUMN\_INDICES(nz) gives the column number for non-zero element nz.

Example of row-column storage scheme on a NxN matrix (N=6). Here the sparsity is 8/36 or 22%

|                 |                    |
|-----------------|--------------------|
| GX 1 2 3 4 5 6  |                    |
| <hr/>           |                    |
| 1   0 A 0 B 0 0 | DATA (nz)          |
| 2   0 0 C 0 0 0 | A B C D E F G H    |
| 3   0 0 0 0 D E | ROW_INDICES (i)    |
| 4   F 0 0 0 0 0 | 1 1 2 3 3 4 5 6    |
| 5   0 0 G 0 0 0 | COLUMN_INDICES (i) |
| 6   0 0 0 0 0 H | 2 4 3 5 6 1 3 6    |

Definition in file [matrix\\_vector.f90](#).

## 8.48 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/mesh\_routines.f90 File Reference

### Id

[mesh\\_routines.f90](#) 181 2008-10-26 18:59:43Z chrispbradley

### Classes

- interface [MESH\\_ROUTINES::MESH\\_NUMBER\\_OF\\_COMPONENTS\\_SET](#)  
*Sets/changes the number of mesh components for a mesh.*
- interface [MESH\\_ROUTINES::MESH\\_NUMBER\\_OF\\_ELEMENTS\\_SET](#)  
*Sets/changes the number of elements for a mesh.*

### Namespaces

- namespace [MESH\\_ROUTINES](#)  
*This module handles all mesh (node and element) routines.*

### Functions

- subroutine [MESH\\_ROUTINES::DECOMPOSITION\\_CREATE\\_FINISH](#) (MESH, DECOMPOSITION, ERR, ERROR,\*)  
*Finishes the creation of a domain decomposition on a given mesh.*
- subroutine [MESH\\_ROUTINES::DECOMPOSITION\\_CREATE\\_START](#) (USER\_NUMBER, MESH, DECOMPOSITION, ERR, ERROR,\*)  
*Starts the creation of a domain decomposition for a given mesh.*
- subroutine [MESH\\_ROUTINES::DECOMPOSITION\\_DESTROY](#) (USER\_NUMBER, MESH, ERR, ERROR,\*)  
*Destroys a domain decomposition identified by a user number and deallocates all memory.*
- subroutine [MESH\\_ROUTINES::DECOMPOSITION\\_ELEMENT\\_DOMAIN\\_CALCULATE](#) (DECOMPOSITION, ERR, ERROR,\*)  
*Calculates the element domains for a decomposition of a mesh.*
- subroutine [MESH\\_ROUTINES::DECOMPOSITION\\_ELEMENT\\_DOMAIN\\_GET](#) (DECOMPOSITION, GLOBAL\_ELEMENT\_NUMBER, DOMAIN\_NUMBER, ERR, ERROR,\*)  
*Gets the domain for a given element in a decomposition of a mesh.*
- subroutine [MESH\\_ROUTINES::DECOMPOSITION\\_ELEMENT\\_DOMAIN\\_SET](#) (DECOMPOSITION, GLOBAL\_ELEMENT\_NUMBER, DOMAIN\_NUMBER, ERR, ERROR,\*)  
*Sets the domain for a given element in a decomposition of a mesh.*
- subroutine [MESH\\_ROUTINES::DECOMPOSITION\\_MESH\\_COMPONENT\\_NUMBER\\_GET](#) (DECOMPOSITION, MESH\_COMPONENT\_NUMBER, ERR, ERROR,\*)

*Gets the mesh component number which will be used for the decomposition of a mesh.*

- subroutine **MESH\_ROUTINES::DECOMPOSITION\_MESH\_COMPONENT\_NUMBER\_SET** (DECOMPOSITION, MESH\_COMPONENT\_NUMBER, ERR, ERROR,\*)

*Sets/changes the mesh component number which will be used for the decomposition of a mesh.*

- subroutine **MESH\_ROUTINES::DECOMPOSITION\_NUMBER\_OF\_DOMAINS\_GET** (DECOMPOSITION, NUMBER\_OF\_DOMAINS, ERR, ERROR,\*)

*Gets the number of domains for a decomposition.*

- subroutine **MESH\_ROUTINES::DECOMPOSITION\_NUMBER\_OF\_DOMAINS\_SET** (DECOMPOSITION, NUMBER\_OF\_DOMAINS, ERR, ERROR,\*)

*Sets/changes the number of domains for a decomposition.*

- subroutine **MESH\_ROUTINES::DOMAIN\_MAPPINGS\_NODES\_FINALISE** (DOMAIN\_MAPPINGS, ERR, ERROR,\*)

*Finalises the node mapping in the given domain mappings.*

- subroutine **MESH\_ROUTINES::DOMAIN\_MAPPINGS\_NODES\_INITIALISE** (DOMAIN\_MAPPINGS, ERR, ERROR,\*)

*Initialises the node mapping in the given domain mapping.*

- subroutine **MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_CALCULATE** (TOPOLOGY, ERR, ERROR,\*)

*Calculates the domain topology.*

- subroutine **MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_INITIALISE\_FROM\_MESH** (DOMAIN, ERR, ERROR,\*)

*Initialises the local domain topology from the mesh topology.*

- subroutine **MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_DOFS\_FINALISE** (TOPOLOGY, ERR, ERROR,\*)

*Finalises the dofs in the given domain topology.*

- subroutine **MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_DOFS\_INITIALISE** (TOPOLOGY, ERR, ERROR,\*)

*Initialises the dofs data structures for a domain topology.*

- subroutine **MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_ELEMENT\_FINALISE** (ELEMENT, ERR, ERROR,\*)

*Finalises the given domain topology element.*

- subroutine **MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_ELEMENT\_INITIALISE** (ELEMENT, ERR, ERROR,\*)

*Initialises the given domain topology element.*

- subroutine **MESH\_ROUTINES::DOMAIN\_TOPOLOGY\_ELEMENTS\_FINALISE** (TOPOLOGY, ERR, ERROR,\*)

*Finalises the elements in the given domain topology.*

- subroutine [MESH\\_ROUTINES::DOMAIN\\_TOPOLOGY\\_ELEMENTS\\_INITIALISE](#) (TOPOLOGY, ERR, ERROR,\*)
 

*Initialises the element data structures for a domain topology.*
- subroutine [MESH\\_ROUTINES::DOMAIN\\_TOPOLOGY\\_FINALISE](#) (DOMAIN, ERR, ERROR,\*)
 

*Finalises the topology in the given domain.*
- subroutine [MESH\\_ROUTINES::DOMAIN\\_TOPOLOGY\\_INITIALISE](#) (DOMAIN, ERR, ERROR,\*)
 

*Initialises the topology for a given domain.*
- subroutine [MESH\\_ROUTINES::DOMAIN\\_TOPOLOGY\\_LINE\\_FINALISE](#) (LINE, ERR, ERROR,\*)
 

*Finalises a line in the given domain topology and deallocates all memory.*
- subroutine [MESH\\_ROUTINES::DOMAIN\\_TOPOLOGY\\_LINE\\_INITIALISE](#) (LINE, ERR, ERROR,\*)
 

*Initialises the line data structure for a domain topology line.*
- subroutine [MESH\\_ROUTINES::DOMAIN\\_TOPOLOGY\\_LINES\\_FINALISE](#) (TOPOLOGY, ERR, ERROR,\*)
 

*Finalises the lines in the given domain topology.*
- subroutine [MESH\\_ROUTINES::DOMAIN\\_TOPOLOGY\\_LINES\\_INITIALISE](#) (TOPOLOGY, ERR, ERROR,\*)
 

*Initialises the line data structures for a domain topology.*
- subroutine [MESH\\_ROUTINES::DOMAIN\\_TOPOLOGY\\_NODE\\_FINALISE](#) (NODE, ERR, ERROR,\*)
 

*Finalises the given domain topology node and deallocates all memory.*
- subroutine [MESH\\_ROUTINES::DOMAIN\\_TOPOLOGY\\_NODE\\_INITIALISE](#) (NODE, ERR, ERROR,\*)
 

*Initialises the given domain topology node.*
- subroutine [MESH\\_ROUTINES::DOMAIN\\_TOPOLOGY\\_NODES\\_FINALISE](#) (TOPOLOGY, ERR, ERROR,\*)
 

*Finalises the nodees in the given domain topology.*
- subroutine [MESH\\_ROUTINES::DOMAIN\\_TOPOLOGY\\_NODES\\_INITIALISE](#) (TOPOLOGY, ERR, ERROR,\*)
 

*Initialises the nodes data structures for a domain topology.*
- subroutine [MESH\\_ROUTINES::DOMAIN\\_TOPOLOGY\\_NODES\\_SURROUNDING\\_ELEMENTS\\_CALCULATE](#) (TOPOLOGY, ERR, ERROR,\*)
 

*Calculates the element numbers surrounding a node for a domain.*
- subroutine [MESH\\_ROUTINES::MESH\\_CREATE\\_FINISH](#) (REGION, MESH, ERR, ERROR,\*)
 

*Finishes the process of creating a mesh on a region.*

- subroutine **MESH\_ROUTINES::MESH\_CREATE\_START** (USER\_NUMBER, REGION, NUMBER\_OF\_DIMENSIONS, MESH, ERR, ERROR,\*)
 

*Starts the process of creating a mesh defined by a user number with the specified NUMBER\_OF\_DIMENSIONS in the region identified by REGION.*
- subroutine **MESH\_ROUTINES::MESH\_DESTROY** (USER\_NUMBER, REGION, ERR, ERROR,\*)
 

*Destroys the mesh identified by a user number on the given region and deallocates all memory.*
- subroutine **MESH\_ROUTINES::MESH\_FINALISE** (MESH, ERR, ERROR,\*)
 

*Finalises a mesh and deallocates all memory.*
- subroutine **MESH\_ROUTINES::MESH\_INITIALISE** (MESH, ERR, ERROR,\*)
 

*Initialises a mesh.*
- INTEGER(INTG) **MESH\_ROUTINES::MESH\_NUMBER\_OF\_COMPONENTS\_GET** (MESH, ERR, ERROR)
 

*Gets the number of mesh components for a mesh identified by a pointer.*
- subroutine **MESH\_ROUTINES::MESH\_NUMBER\_OF\_COMPONENTS\_SET\_NUMBER** (USER\_NUMBER, REGION, NUMBER\_OF\_COMPONENTS, ERR, ERROR,\*)
 

*Changes/sets the number of mesh components for a mesh identified by a given user number on a region.*
- subroutine **MESH\_ROUTINES::MESH\_NUMBER\_OF\_COMPONENTS\_SET\_PTR** (MESH, NUMBER\_OF\_COMPONENTS, ERR, ERROR,\*)
 

*Changes/sets the number of mesh components for a mesh identified by a pointer.*
- INTEGER(INTG) **MESH\_ROUTINES::MESH\_NUMBER\_OF\_ELEMENTS\_GET** (MESH, ERR, ERROR)
 

*Gets the number of elements for a mesh identified by a pointer.*
- subroutine **MESH\_ROUTINES::MESH\_NUMBER\_OF\_ELEMENTS\_SET\_NUMBER** (USER\_NUMBER, REGION, NUMBER\_OF\_ELEMENTS, ERR, ERROR,\*)
 

*Changes/sets the number of elements for a mesh identified by a given user number on a region.*
- subroutine **MESH\_ROUTINES::MESH\_NUMBER\_OF\_ELEMENTS\_SET\_PTR** (MESH, NUMBER\_OF\_ELEMENTS, ERR, ERROR,\*)
 

*Changes/sets the number of elements for a mesh identified by a pointer.*
- subroutine **MESH\_ROUTINES::MESH\_TOPOLOGY\_CALCULATE** (TOPOLOGY, ERR, ERROR,\*)
 

*Calculates the mesh topology.*
- subroutine **MESH\_ROUTINES::MESH\_TOPOLOGY\_DOFS\_CALCULATE** (TOPOLOGY, ERR, ERROR,\*)
 

*Calculates the degrees-of-freedom for a mesh topology.*
- subroutine **MESH\_ROUTINES::MESH\_TOPOLOGY\_DOFS\_FINALISE** (TOPOLOGY, ERR, ERROR,\*)
 

*Finalises the dof data structures for a mesh topology and deallocates any memory.*

- subroutine [MESH\\_ROUTINES::MESH\\_TOPOLOGY\\_DOFS\\_INITIALISE](#) (TOPOLOGY, ERR, ERROR,\*)
 

*Initialises the dofs in a given mesh topology.*
- subroutine [MESH\\_ROUTINES::MESH\\_TOPOLOGY\\_ELEMENTS\\_BASIS\\_SET](#) (GLOBAL\_NUMBER, ELEMENTS, BASIS, ERR, ERROR,\*)
 

*Changes/sets the basis for a global element identified by a given global number.*
- subroutine [MESH\\_ROUTINES::MESH\\_TOPOLOGY\\_ELEMENTS\\_CREATE\\_FINISH](#) (MESH, MESH\_COMPONENT\_NUMBER, ERR, ERROR,\*)
 

*Finishes the process of creating elements for a specified mesh component in a mesh topology.*
- subroutine [MESH\\_ROUTINES::MESH\\_TOPOLOGY\\_ELEMENTS\\_CREATE\\_START](#) (MESH, MESH\_COMPONENT\_NUMBER, BASIS, ELEMENTS, ERR, ERROR,\*)
 

*Starts the process of creating elements in the mesh component identified by MESH and component\_idx. The elements will be created with a default basis of BASIS. ELEMENTS is the returned pointer to the MESH\_ELEMENTS data structure.*
- subroutine [MESH\\_ROUTINES::MESH\\_TOPOLOGY\\_ELEMENTS\\_DESTROY](#) (TOPOLOGY, ERR, ERROR,\*)
 

*Destroys the elements in a mesh topology.*
- subroutine [MESH\\_ROUTINES::MESH\\_TOPOLOGY\\_ELEMENT\\_FINALISE](#) (ELEMENT, ERR, ERROR,\*)
 

*Finalises the given mesh topology element.*
- subroutine [MESH\\_ROUTINES::MESH\\_TOPOLOGY\\_ELEMENT\\_INITIALISE](#) (ELEMENT, ERR, ERROR,\*)
 

*Initialises the given mesh topology element.*
- subroutine [MESH\\_ROUTINES::MESH\\_TOPOLOGY\\_ELEMENTS\\_ELEMENT\\_BASIS\\_GET](#) (GLOBAL\_NUMBER, ELEMENTS, BASIS, ERR, ERROR,\*)
 

*Gets the basis for a mesh element identified by a given global number.*
- subroutine [MESH\\_ROUTINES::MESH\\_TOPOLOGY\\_ELEMENTS\\_ELEMENT\\_BASIS\\_SET](#) (GLOBAL\_NUMBER, ELEMENTS, BASIS, ERR, ERROR,\*)
 

*Changes/sets the basis for a mesh element identified by a given global number.*
- subroutine [MESH\\_ROUTINES::MESH\\_TOPOLOGY\\_ELEMENTS\\_ELEMENT\\_NODES\\_GET](#) (GLOBAL\_NUMBER, ELEMENTS, USER\_ELEMENT\_NODES, ERR, ERROR,\*)
 

*Gets the element nodes for a mesh element identified by a given global number.*
- subroutine [MESH\\_ROUTINES::MESH\\_TOPOLOGY\\_ELEMENTS\\_ELEMENT\\_NODES\\_SET](#) (GLOBAL\_NUMBER, ELEMENTS, USER\_ELEMENT\_NODES, ERR, ERROR,\*)
 

*Changes/sets the element nodes for a mesh element identified by a given global number.*
- subroutine [MESH\\_ROUTINES::MESH\\_TOPOLOGY\\_ELEMENTS\\_ADJACENT\\_ELEMENTS\\_CALCULATE](#) (TOPOLOGY, ERR, ERROR,\*)
 

*Calculates the element numbers surrounding an element in a mesh topology.*

- subroutine **MESH\_ROUTINES::MESH\_TOPOLOGY\_ELEMENTS\_FINALISE** (TOPOLOGY, ERR, ERROR,\*)
 

*Finalises the elements data structures for a mesh topology and deallocates any memory.*
- subroutine **MESH\_ROUTINES::MESH\_TOPOLOGY\_ELEMENTS\_INITIALISE** (TOPOLOGY, ERR, ERROR,\*)
 

*Initialises the elements in a given mesh topology.*
- subroutine **MESH\_ROUTINES::MESH\_TOPOLOGY\_ELEMENTS\_NUMBER\_GET** (GLOBAL\_NUMBER, USER\_NUMBER, ELEMENTS, ERR, ERROR,\*)
 

*Gets the user number for a global element identified by a given global number.*
- subroutine **MESH\_ROUTINES::MESH\_TOPOLOGY\_ELEMENTS\_NUMBER\_SET** (GLOBAL\_NUMBER, USER\_NUMBER, ELEMENTS, ERR, ERROR,\*)
 

*Changes/sets the user number for a global element identified by a given global number.*
- subroutine **MESH\_ROUTINES::MESH\_TOPOLOGY\_FINALISE** (MESH, ERR, ERROR,\*)
 

*Finalises the topology in the given mesh.*
- subroutine **MESH\_ROUTINES::MESH\_TOPOLOGY\_INITIALISE** (MESH, ERR, ERROR,\*)
 

*Initialises the topology for a given mesh.*
- subroutine **MESH\_ROUTINES::MESH\_TOPOLOGY\_NODE\_FINALISE** (NODE, ERR, ERROR,\*)
 

*Finalises the given mesh topology node.*
- subroutine **MESH\_ROUTINES::MESH\_TOPOLOGY\_NODE\_INITIALISE** (NODE, ERR, ERROR,\*)
 

*Initialises the given mesh topology node.*
- subroutine **MESH\_ROUTINES::MESH\_TOPOLOGY\_NODES\_CALCULATE** (TOPOLOGY, ERR, ERROR,\*)
 

*Calculates the nodes used the mesh identified by a given mesh topology.*
- subroutine **MESH\_ROUTINES::MESH\_TOPOLOGY\_NODES\_DERIVATIVES\_CALCULATE** (TOPOLOGY, ERR, ERROR,\*)
 

*Calculates the number of derivatives at each node in a topology.*
- subroutine **MESH\_ROUTINES::MESH\_TOPOLOGY\_NODES\_SURROUNDING\_ELEMENTS\_CALCULATE** (TOPOLOGY, ERR, ERROR,\*)
 

*Calculates the element numbers surrounding a node for a mesh.*
- subroutine **MESH\_ROUTINES::MESH\_TOPOLOGY\_NODES\_FINALISE** (TOPOLOGY, ERR, ERROR,\*)
 

*Finalises the nodes data structures for a mesh topology and deallocates any memory.*
- subroutine **MESH\_ROUTINES::MESH\_TOPOLOGY\_NODES\_INITIALISE** (TOPOLOGY, ERR, ERROR,\*)
 

*Initialises the nodes in a given mesh topology.*

- subroutine [MESH\\_ROUTINES::MESH\\_USER\\_NUMBER\\_FIND](#) (USER\_NUMBER, REGION, MESH, ERR, ERROR,\*)
 

*Finds and returns in MESH a pointer to the mesh identified by USER\_NUMBER in the given REGION. If no mesh with that number exists MESH is left nullified.*
- subroutine [MESH\\_ROUTINES::MESHES\\_FINALISE](#) (REGION, ERR, ERROR,\*)
 

*Finalises the meshes in the given region.*
- subroutine [MESH\\_ROUTINES::MESHES\\_INITIALISE](#) (REGION, ERR, ERROR,\*)
 

*Initialises the meshes for the given region.*

## Variables

- INTEGER(INTG), parameter [MESH\\_ROUTINES::DECOMPOSITION\\_ALL\\_TYPE](#) = 1
 

*The decomposition contains all elements.*
- INTEGER(INTG), parameter [MESH\\_ROUTINES::DECOMPOSITION\\_CALCULATED\\_TYPE](#) = 2
 

*The element decomposition is calculated by graph partitioning.*
- INTEGER(INTG), parameter [MESH\\_ROUTINES::DECOMPOSITION\\_USER\\_DEFINED\\_TYPE](#) = 3
 

*The user will set the element decomposition.*

### 8.48.1 Detailed Description

#### Id

[mesh\\_routines.f90](#) 181 2008-10-26 18:59:43Z chrisbradley

#### Author:

Chris Bradley This module handles all mesh (node and element) routines.

### 8.48.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [mesh\\_routines.f90](#).

## 8.49 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/node\_routines.f90 File Reference

### Id

[node\\_routines.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

### Namespaces

- namespace [NODE\\_ROUTINES](#)  
*This module handles all node routines.*

### Functions

- subroutine [NODE\\_ROUTINES::NODE\\_CHECK\\_EXISTS](#) (USER\_NUMBER, REGION, NODE\_EXISTS, GLOBAL\_NUMBER, ERR, ERROR,\*)
- subroutine [NODE\\_ROUTINES::NODE\\_DESTROY](#) (NODE, ERR, ERROR,\*)
- subroutine [NODE\\_ROUTINES::NODE\\_INITIAL\\_POSITION\\_SET](#) (GLOBAL\_NUMBER, INITIAL\_POSITION, NODES, ERR, ERROR,\*)
- subroutine [NODE\\_ROUTINES::NODE\\_NUMBER\\_SET](#) (GLOBAL\_NUMBER, USER\_NUMBER, NODES, ERR, ERROR,\*)
- subroutine [NODE\\_ROUTINES::NODES\\_CREATE\\_FINISH](#) (REGION, ERR, ERROR,\*)
- subroutine [NODE\\_ROUTINES::NODES\\_CREATE\\_START](#) (NUMBER\_OF\_NODES, REGION, NODES, ERR, ERROR,\*)
- subroutine [NODE\\_ROUTINES::NODES\\_FINALISE](#) (REGION, ERR, ERROR,\*)
- subroutine [NODE\\_ROUTINES::NODES\\_INITIALISE](#) (REGION, ERR, ERROR,\*)

### 8.49.1 Detailed Description

#### Id

[node\\_routines.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

#### Author:

Chris Bradley This module handles all node routines.

### 8.49.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University

of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [node\\_routines.f90](#).

## 8.50 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/problem\_- constants.f90 File Reference

### Id

[problem\\_constants.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

### Namespaces

- namespace PROBLEM\_CONSTANTS

*This module handles all problem wide constants.*

### Variables

- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_NO\_CLASS = 0
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_ELASTICITY\_CLASS = 1
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_FLUID\_MECHANICS\_CLASS = 2
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_ELECTROMAGNETICS\_CLASS = 3
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_CLASSICAL\_FIELD\_CLASS = 4
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_MODAL\_CLASS = 5
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_FITTING\_CLASS = 6
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_OPTIMISATION\_CLASS = 7
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_NO\_TYPE = 0
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_LINEAR\_ELASTICITY\_TYPE = 1
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEMFINITE\_ELASTICITY\_TYPE = 2
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_STOKES\_FLUID\_TYPE = 1
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_NAVIER\_STOKES\_FLUID\_TYPE = 2
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_ELECTROSTATIC\_TYPE = 1
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_MAGNETOSTATIC\_TYPE = 2
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_MAXWELLS\_EQUATIONS\_TYPE = 3
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_LAPLACE\_EQUATION\_TYPE = 1
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_POISSON\_EQUATION\_TYPE = 2
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_HELMHOLTZ\_EQUATION\_TYPE = 3
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_WAVE\_EQUATION\_TYPE = 4

- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_DIFFUSION\_-  
**EQUATION\_TYPE** = 5
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_ADVECTION\_-  
**DIFFUSION\_EQUATION\_TYPE** = 6
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_REACTION\_-  
**DIFFUSION\_EQUATION\_TYPE** = 7
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_BIHAMONIC\_-  
**EQUATION\_TYPE** = 8
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_LINEAR\_ELASTIC\_-  
**MODAL\_TYPE** = 1
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_NO\_SUBTYPE = 0
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_STANDARD\_LAPLACE\_-  
**SUBTYPE** = 1
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_GENERALISED\_-  
**LAPLACE\_SUBTYPE** = 2
- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_INITIAL\_TYPE = 1

*Initial setup for a problem.*

- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_CONTROL\_TYPE = 2

*Solver setup for a problem.*

- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_SOLUTION\_-  
**TYPE** = 3

*Solution parameters setup for a problem.*

- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_SOLVER\_TYPE = 4

*Solver setup for a problem.*

- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_START\_ACTION = 1

*Start setup action.*

- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_FINISH\_ACTION = 2

*Finish setup action.*

- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM\_SETUP\_DO\_ACTION = 3

*Do setup action.*

- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM SOLUTION\_NO\_-  
**OUTPUT** = 0

*No output.*

- INTEGER(INTG), parameter PROBLEM\_CONSTANTS::PROBLEM SOLUTION\_TIMING\_-  
**OUTPUT** = 1

*Timing information output.*

- INTEGER(INTG), parameter **PROBLEM\_CONSTANTS::PROBLEM SOLUTION MATRIX - OUTPUT = 2**  
*All below and solution matrices output.*
- INTEGER(INTG), parameter **PROBLEM\_CONSTANTS::PROBLEM SOLUTION LINEAR = 1**  
*The problem solution is linear.*
- INTEGER(INTG), parameter **PROBLEM\_CONSTANTS::PROBLEM SOLUTION NONLINEAR = 2**  
*The problem solution is nonlinear.*
- INTEGER(INTG), parameter **PROBLEM\_CONSTANTS::PROBLEM SOLVER NO\_OUTPUT = 0**  
*No output.*
- INTEGER(INTG), parameter **PROBLEM\_CONSTANTS::PROBLEM SOLVER TIMING - OUTPUT = 1**  
*Timing information output.*
- INTEGER(INTG), parameter **PROBLEM\_CONSTANTS::PROBLEM SOLVER SOLVER - OUTPUT = 2**  
*All below and solver output.*
- INTEGER(INTG), parameter **PROBLEM\_CONSTANTS::PROBLEM SOLVER SPARSE - MATRICES = 1**  
*Use sparse solver matrices.*
- INTEGER(INTG), parameter **PROBLEM\_CONSTANTS::PROBLEM SOLVER FULL - MATRICES = 2**  
*Use fully populated solver matrices.*

### 8.50.1 Detailed Description

#### Id

[problem\\_constants.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

#### Author:

Chris Bradley This module handles all problem wide constants.

### 8.50.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [problem\\_constants.f90](#).

## 8.51 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/problem\_routines.f90 File Reference

### Id

[problem\\_routines.f90](#) 178 2008-10-26 13:07:57Z chrisbradley

### Classes

- interface [PROBLEM\\_ROUTINES::PROBLEM\\_DESTROY](#)
- interface [PROBLEM\\_ROUTINES::PROBLEM\\_SPECIFICATION\\_GET](#)
- interface [PROBLEM\\_ROUTINES::PROBLEM\\_SPECIFICATION\\_SET](#)

### Namespaces

- namespace [PROBLEM\\_ROUTINES](#)

*This module handles all problem routines.*

### Functions

- subroutine [PROBLEM\\_ROUTINES::PROBLEM\\_CREATE\\_FINISH](#) (PROBLEM, ERR, ERROR,\*)

*Finishes the process of creating a problem.*
- subroutine [PROBLEM\\_ROUTINES::PROBLEM\\_CREATE\\_START](#) (USER\_NUMBER, PROBLEM, ERR, ERROR,\*)

*Starts the process of creating a problem defined by USER\_NUMBER.*
- subroutine [PROBLEM\\_ROUTINES::PROBLEM\\_DESTROY\\_NUMBER](#) (USER\_NUMBER, ERR, ERROR,\*)

*Destroys a problem identified by a user number.*
- subroutine [PROBLEM\\_ROUTINES::PROBLEM\\_DESTROY\\_PTR](#) (PROBLEM, ERR, ERROR,\*)

*Destroys a problem identified by a pointer.*
- subroutine [PROBLEM\\_ROUTINES::PROBLEM\\_FINALISE](#) (PROBLEM, ERR, ERROR,\*)

*Finalise the problem and deallocate all memory.*
- subroutine [PROBLEM\\_ROUTINES::PROBLEM\\_INITIALISE](#) (PROBLEM, ERR, ERROR,\*)

*Initialises a problem.*
- subroutine [PROBLEM\\_ROUTINES::PROBLEM\\_CONTROL\\_CREATE\\_FINISH](#) (PROBLEM, ERR, ERROR,\*)

*Finish the creation of the control for the problem.*
- subroutine [PROBLEM\\_ROUTINES::PROBLEM\\_CONTROL\\_CREATE\\_START](#) (PROBLEM, ERR, ERROR,\*)

*Start the creation of a problem control for a problem.*

- subroutine **PROBLEM\_ROUTINES::PROBLEM\_CONTROL\_DESTROY** (PROBLEM, ERR, ERROR,\*)

*Destroy the control for a problem.*

- subroutine **PROBLEM\_ROUTINES::PROBLEM\_CONTROL\_FINALISE** (CONTROL, ERR, ERROR,\*)

*Finalise the control for a problem and deallocate all memory.*

- subroutine **PROBLEM\_ROUTINES::PROBLEM\_CONTROL\_INITIALISE** (PROBLEM, ERR, ERROR,\*)

*Initialise the control for a problem.*

- subroutine **PROBLEM\_ROUTINES::PROBLEM SOLUTION JACOBIAN EVALUATE** (SOLUTION, ERR, ERROR,\*)

*Evaluates the Jacobian for a nonlinear problem solution.*

- subroutine **PROBLEM\_ROUTINES::PROBLEM SOLUTION RESIDUAL EVALUATE** (SOLUTION, ERR, ERROR,\*)

*Evaluates the residual for a nonlinear problem solution.*

- subroutine **PROBLEM\_ROUTINES::PROBLEM SETUP** (PROBLEM, SETUP\_TYPE, ACTION\_TYPE, ERR, ERROR,\*)

*Sets up the specifics for a problem.*

- subroutine **PROBLEM\_ROUTINES::PROBLEM SOLVE** (PROBLEM, ERR, ERROR,\*)

*Solves a problem.*

- subroutine **PROBLEM\_ROUTINES::PROBLEM SOLUTION SOLVE** (SOLUTION, ERR, ERROR,\*)

*Solves a solution.*

- subroutine **PROBLEM\_ROUTINES::PROBLEM SOLUTIONS CREATE FINISH** (PROBLEM, ERR, ERROR,\*)

*Finish the creation of solutions for a problem.*

- subroutine **PROBLEM\_ROUTINES::PROBLEM SOLUTIONS CREATE START** (PROBLEM, ERR, ERROR,\*)

*Start the creation of a solution for the problem.*

- subroutine **PROBLEM\_ROUTINES::PROBLEM SOLUTION EQUATIONS SET ADD** (PROBLEM, SOLUTION\_INDEX, EQUATIONS\_SET, EQUATIONS\_SET\_INDEX, ERR, ERROR,\*)

*Adds an equations set to a problem solution.*

- subroutine **PROBLEM\_ROUTINES::PROBLEM SOLUTION FINALISE** (SOLUTION, ERR, ERROR,\*)

*Finalises a solution and deallocates all memory.*

- subroutine **PROBLEM\_ROUTINES::PROBLEM SOLUTION INITIALISE** (SOLUTION, ERR, ERROR,\*)

*Initialises the solution for a problem.*

- subroutine `PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_FINALISE` (PROBLEM, ERR, ERROR,\*)

*Finalises the solutions for a problem and deallocates all memory.*

- subroutine `PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_INITIALISE` (PROBLEM, ERR, ERROR,\*)

*Initialises the solutions for a problem.*

- subroutine `PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_FINISH` (PROBLEM, ERR, ERROR,\*)

*Finish the creation of the solver for the problem solutions.*

- subroutine `PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_START` (PROBLEM, ERR, ERROR,\*)

*Start the creation of a problem solver for a problem.*

- subroutine `PROBLEM_ROUTINES::PROBLEM_SOLVER_DESTROY` (PROBLEM, ERR, ERROR,\*)

*Destroy the solvers for a problem.*

- subroutine `PROBLEM_ROUTINES::PROBLEM_SOLVER_GET` (PROBLEM, SOLUTION\_INDEX, SOLVER, ERR, ERROR,\*)

*Returns a pointer to the solver for a solution on a problem.*

- subroutine `PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_GET_NUMBER` (USER\_NUMBER, PROBLEM\_CLASS, PROBLEM\_EQUATION\_TYPE, PROBLEM\_SUBTYPE, ERR, ERROR,\*)

*Gets the problem specification i.e., problem class, type and subtype for a problem identified by a user number.*

- subroutine `PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_GET_PTR` (PROBLEM, PROBLEM\_CLASS, PROBLEM\_EQUATION\_TYPE, PROBLEM\_SUBTYPE, ERR, ERROR,\*)

*Gets the problem specification i.e., problem class, type and subtype for a problem identified by a pointer.*

- subroutine `PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_NUMBER` (USER\_NUMBER, PROBLEM\_CLASS, PROBLEM\_EQUATION\_TYPE, PROBLEM\_SUBTYPE, ERR, ERROR,\*)

*Sets/changes the problem specification i.e., problem class, type and subtype for a problem identified by a user number.*

- subroutine `PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_PTR` (PROBLEM, PROBLEM\_CLASS, PROBLEM\_EQUATION\_TYPE, PROBLEM\_SUBTYPE, ERR, ERROR,\*)

*Sets/changes the problem specification i.e., problem class, type and subtype for a problem identified by a pointer.*

- subroutine `PROBLEM_ROUTINES::PROBLEM_USER_NUMBER_FIND` (USER\_NUMBER, PROBLEM, ERR, ERROR,\*)

*Finds and returns in PROBLEM a pointer to the problem identified by USER\_NUMBER. If no problem with that USER\_NUMBER exists PROBLEM is left nullified.*

- subroutine **PROBLEM\_ROUTINES::PROBLEMS\_FINALISE** (ERR, ERROR,\*)  
*Finalises all problems and deallocates all memory.*
- subroutine **PROBLEM\_ROUTINES::PROBLEMS\_INITIALISE** (ERR, ERROR,\*)  
*Initialises all problems.*
- subroutine **PROBLEM\_SOLUTION\_JACOBIAN\_EVALUATE\_PETSC** (SNES, A, B, FLAG, CTX)  
*Called from the PETSc SNES solvers to evaluate the Jacobian for a Newton like nonlinear solver.*
- subroutine **PROBLEM\_SOLUTION\_RESIDUAL\_EVALUATE\_PETSC** (SNES, X, F, CTX)  
*Called from the PETSc SNES solvers to evaluate the residual for a Newton like nonlinear solver.*

## Variables

- INTEGER(INTG), parameter **PROBLEM\_ROUTINES::NUMBER\_OF\_PROBLEM\_LINEARITIES** = 3  
*The number of problem linearity types defined.*
- INTEGER(INTG), parameter **PROBLEM\_ROUTINES::PROBLEM\_LINEAR** = 1  
*The problem is linear.*
- INTEGER(INTG), parameter **PROBLEM\_ROUTINES::PROBLEM\_NONLINEAR** = 2  
*The problem is non-linear.*
- INTEGER(INTG), parameter **PROBLEM\_ROUTINES::PROBLEM\_NONLINEAR\_BCS** = 3  
*The problem has non-linear boundary conditions.*
- INTEGER(INTG), parameter **PROBLEM\_ROUTINES::NUMBER\_OF\_PROBLEM\_TIME\_TYPES** = 3  
*The number of problem time dependence types defined.*
- INTEGER(INTG), parameter **PROBLEM\_ROUTINES::PROBLEM\_STATIC** = 1  
*The problem is static and has no time dependence.*
- INTEGER(INTG), parameter **PROBLEM\_ROUTINES::PROBLEM\_DYNAMIC** = 2  
*The problem is dynamic.*
- INTEGER(INTG), parameter **PROBLEM\_ROUTINES::PROBLEM\_QUASISTATIC** = 3  
*The problem is quasi-static.*
- TYPE(**PROBLEMS\_TYPE**), target **PROBLEM\_ROUTINES::PROBLEMS**

### 8.51.1 Detailed Description

#### Id

[problem\\_routines.f90](#) 178 2008-10-26 13:07:57Z chrisbradley

#### Author:

Chris Bradley This module handles all problem routines.

### 8.51.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

#### Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [problem\\_routines.f90](#).

### 8.51.3 Function Documentation

#### 8.51.3.1 subroutine PROBLEM SOLUTION JACOBIAN EVALUATE PETSC

```
(INTEGER(INTG),dimension(*) SNES, INTEGER(INTG),dimension(*)
A, INTEGER(INTG),dimension(*) B, INTEGER(INTG) FLAG,
TYPE(SOLUTION_TYPE),pointer CTX) [private]
```

Called from the PETSc SNES solvers to evaluate the Jacobian for a Newton like nonlinear solver.

#### Parameters:

**SNES** The PETSc SNES type

**A** The PETSc A Mat type

**B** The PETSc B Mat type

**FLAG** The PETSC MatStructure flag

*CTX* The passed through context

Definition at line 1607 of file problem\_routines.f90.

References PROBLEM\_ROUTINES::PROBLEM SOLUTION JACOBIAN EVALUATE().

Here is the call graph for this function:

**8.51.3.2 subroutine PROBLEM SOLUTION RESIDUAL EVALUATE PETSC**  
**(INTEGER(INTG),dimension(\*) SNES, INTEGER(INTG),dimension(\*) X,**  
**INTEGER(INTG),dimension(\*) F, TYPE(SOLUTION\_TYPE),pointer CTX)**

Called from the PETSc SNES solvers to evaluate the residual for a Newton like nonlinear solver.

**Parameters:**

*SNES* The PETSc SNES type

*X* The PETSc X Vec type

*F* The PETSc F Vec type

*CTX* The passed through context

Definition at line 1636 of file problem\_routines.f90.

References PROBLEM\_ROUTINES::PROBLEM SOLUTION RESIDUAL EVALUATE().

Referenced by SOLVER\_ROUTINES::SOLVER NONLINEAR LINESEARCH CREATE FINISH(), and SOLVER\_ROUTINES::SOLVER NONLINEAR TRUSTREGION CREATE FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

## 8.52 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/region\_routines.f90 File Reference

### Id

[region\\_routines.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

### Classes

- interface [REGION\\_ROUTINES::REGION\\_COORDINATE\\_SYSTEM\\_SET](#)
- interface [REGION\\_ROUTINES::REGION\\_LABEL\\_SET](#)

### Namespaces

- namespace [REGION\\_ROUTINES](#)

*This module contains all region routines.*

### Functions

- TYPE(COORDINATE\_SYSTEM\_TYPE)      [REGION\\_ROUTINES::REGION\\_COORDINATE\\_SYSTEM\\_GET](#) (REGION, ERR, ERROR)
- subroutine      [REGION\\_ROUTINES::REGION\\_COORDINATE\\_SYSTEM\\_SET\\_NUMBER](#) (USER\_NUMBER, COORDINATE\_SYSTEM, ERR, ERROR,\*)
- subroutine      [REGION\\_ROUTINES::REGION\\_COORDINATE\\_SYSTEM\\_SET\\_PTR](#) (REGION, COORDINATE\_SYSTEM, ERR, ERROR,\*)
- subroutine      [REGION\\_ROUTINES::REGION\\_CREATE\\_FINISH](#) (REGION, ERR, ERROR,\*)
- subroutine      [REGION\\_ROUTINES::REGION\\_CREATE\\_START](#) (USER\_NUMBER, REGION, ERR, ERROR,\*)
- subroutine      [REGION\\_ROUTINES::REGION\\_DESTROY](#) (USER\_NUMBER, ERR, ERROR,\*)
- TYPE(VARYING\_STRING)      [REGION\\_ROUTINES::REGION\\_LABEL\\_GET](#) (REGION, ERR, ERROR)
- subroutine      [REGION\\_ROUTINES::REGION\\_LABEL\\_SET\\_NUMBER](#) (USER\_NUMBER, LABEL, ERR, ERROR,\*)
- subroutine      [REGION\\_ROUTINES::REGION\\_LABEL\\_SET\\_PTR](#) (REGION, LABEL, ERR, ERROR,\*)
- subroutine      [REGION\\_ROUTINES::REGION\\_SUB\\_REGION\\_CREATE\\_START](#) (USER\_NUMBER, PARENT\_REGION, SUB\_REGION, ERR, ERROR,\*)
- subroutine      [REGION\\_ROUTINES::REGION\\_SUB\\_REGION\\_CREATE\\_FINISH](#) (REGION, ERR, ERROR,\*)
- subroutine      [REGION\\_ROUTINES::REGION\\_USER\\_NUMBER\\_FIND](#) (USER\_NUMBER, REGION, ERR, ERROR,\*)
- subroutine      [REGION\\_ROUTINES::REGION\\_USER\\_NUMBER\\_FIND\\_PTR](#) (USER\_NUMBER, REGION, START\_REGION, ERR, ERROR,\*)
- subroutine      [REGION\\_ROUTINES::REGIONS\\_INITIALISE](#) (ERR, ERROR,\*)
- subroutine      [REGION\\_ROUTINES::REGIONS\\_FINALISE](#) (ERR, ERROR,\*)

### Variables

- TYPE(REGION\_TYPE), target [REGION\\_ROUTINES::GLOBAL\\_REGION](#)

### 8.52.1 Detailed Description

#### Id

[region\\_routines.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

#### Author:

Chris Bradley This module contains all region routines.

### 8.52.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [region\\_routines.f90](#).

## 8.53 **d:/Users/tyu011/workspace/OpenCMISS-trunk/src/solution\_- mapping\_routines.f90 File Reference**

## 8.54 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/solver\_- matrices\_routines.f90 File Reference

### Id

[solver\\_matrices\\_routines.f90](#) 178 2008-10-26 13:07:57Z chrispbradley

### Namespaces

- namespace [SOLVER\\_MATRICES\\_ROUTINES](#)  
*This module handles all solver matrix and rhs routines.*

### Functions

- subroutine [SOLVER\\_MATRICES\\_ROUTINES::SOLVER\\_MATRICES\\_CREATE\\_FINISH](#) (SOLVER\_MATRICES, ERR, ERROR,\*)  
*Finishes the process of creating the solver matrices.*
- subroutine [SOLVER\\_MATRICES\\_ROUTINES::SOLVER\\_MATRICES\\_CREATE\\_START](#) (SOLVER, SOLVER\_MATRICES, ERR, ERROR,\*)  
*Starts the process of creating the solver matrices.*
- subroutine [SOLVER\\_MATRICES\\_ROUTINES::SOLVER\\_MATRICES\\_DESTROY](#) (SOLVER\_MATRICES, ERR, ERROR,\*)  
*Destroys the solver matrices.*
- subroutine [SOLVER\\_MATRICES\\_ROUTINES::SOLVER\\_MATRICES\\_FINALISE](#) (SOLVER\_MATRICES, ERR, ERROR,\*)  
*Finalises the solver matrices and deallocates all memory.*
- subroutine [SOLVER\\_MATRICES\\_ROUTINES::SOLVER\\_MATRICES\\_INITIALISE](#) (SOLVER, ERR, ERROR,\*)  
*Initialises the solver matrices for a solver.*
- INTEGER(INTG) [SOLVER\\_MATRICES\\_ROUTINES::SOLVER\\_MATRICES\\_LIBRARY\\_TYPE\\_GET](#) (SOLVER\_MATRICES, ERR, ERROR)  
*Gets the library type for the solver matrices (and vectors).*
- subroutine [SOLVER\\_MATRICES\\_ROUTINES::SOLVER\\_MATRICES\\_LIBRARY\\_TYPE\\_SET](#) (SOLVER\_MATRICES, LIBRARY\_TYPE, ERR, ERROR,\*)  
*Sets the library type for the solver matrices (and vectors).*
- subroutine [SOLVER\\_MATRICES\\_ROUTINES::SOLVER\\_MATRICES\\_OUTPUT](#) (ID, SOLVER\_MATRICES, ERR, ERROR,\*)  
*Outputs the solver matrices.*
- subroutine [SOLVER\\_MATRICES\\_ROUTINES::SOLVER\\_MATRICES\\_STORAGE\\_TYPE\\_GET](#) (SOLVER\_MATRICES, STORAGE\_TYPE, ERR, ERROR,\*)  
*Gets the storage type (sparsity) of the solver matrices.*

- subroutine **SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRICES\_STORAGE\_TYPE\_SET** (SOLVER\_MATRICES, STORAGE\_TYPE, ERR, ERROR,\*)
 

*Sets the storage type (sparsity) of the solver matrices.*
- subroutine **SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRIX\_STRUCTURE\_CALCULATE** (SOLVER\_MATRIX, NUMBER\_OF\_NON\_ZEROS, ROW\_INDICES, COLUMN\_INDICES, ERR, ERROR,\*)
 

*Calculates the structure (sparsity) of the solver matrix from the solution mapping.*
- subroutine **SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRIX\_FINALISE** (SOLVER\_MATRIX, ERR, ERROR,\*)
 

*Finalises a solver matrix and deallocates all memory.*
- subroutine **SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRIX\_FORM** (SOLVER\_MATRIX, ERR, ERROR,\*)
 

*Forms a solver matrix by initialising the structure of the matrix to zero.*
- subroutine **SOLVER\_MATRICES\_ROUTINES::SOLVER\_MATRIX\_INITIALISE** (SOLVER\_MATRICES, MATRIX\_NUMBER, ERR, ERROR,\*)
 

*Initialises a solver matrix.*

### 8.54.1 Detailed Description

#### Id

[solver\\_matrices\\_routines.f90](#) 178 2008-10-26 13:07:57Z chrisbradley

#### Author:

Chris Bradley This module handles all solver matrix and rhs routines.

### 8.54.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the

"LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [solver\\_matrices\\_routines.f90](#).

## 8.55 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/solver\_routines.f90 File Reference

### Id

[solver\\_routines.f90](#) 181 2008-10-26 18:59:43Z chrispbradley

### Namespaces

- namespace [SOLVER\\_ROUTINES](#)

*This module handles all solver routines.*

### Functions

- subroutine [SOLVER\\_ROUTINES::SOLVER\\_CREATE\\_FINISH](#) (SOLVER, ERR, ERROR,\*)
 

*Finishes the process of creating a solver for a problem solution.*
- subroutine [SOLVER\\_ROUTINES::SOLVER\\_CREATE\\_START](#) (SOLUTION, SOLVE\_TYPE, SOLVER, ERR, ERROR,\*)
 

*Starts the process of creating a solver for a problem solution.*
- subroutine [SOLVER\\_ROUTINES::SOLVER\\_DESTROY](#) (SOLVER, ERR, ERROR,\*)
 

*Destroy a problem solver.*
- subroutine [SOLVER\\_ROUTINES::SOLVER\\_EIGENPROBLEM\\_CREATE\\_FINISH](#) (EIGENPROBLEM\_SOLVER, ERR, ERROR,\*)
 

*Finishes the process of creating a eigenproblem solver.*
- subroutine [SOLVER\\_ROUTINES::SOLVER\\_EIGENPROBLEM\\_FINALISE](#) (EIGENPROBLEM\_SOLVER, ERR, ERROR,\*)
 

*Finalise a eigenproblem solver for a problem solver.*
- subroutine [SOLVER\\_ROUTINES::SOLVER\\_EIGENPROBLEM\\_INITIALISE](#) (SOLVER, ERR, ERROR,\*)
 

*Initialise a eigenproblem solver for a problem solver.*
- subroutine [SOLVER\\_ROUTINES::SOLVER\\_EIGENPROBLEM\\_SOLVE](#) (EIGENPROBLEM\_SOLVER, ERR, ERROR,\*)
 

*Solve a eigenproblem solver.*
- subroutine [SOLVER\\_ROUTINES::SOLVER\\_FINALISE](#) (SOLVER, ERR, ERROR,\*)
 

*Finalises a problem solver and deallocates all memory.*
- subroutine [SOLVER\\_ROUTINES::SOLVER\\_INITIALISE](#) (SOLUTION, SOLVE\_TYPE, ERR, ERROR,\*)
 

*Initialise a solver for a problem solution.*
- subroutine [SOLVER\\_ROUTINES::SOLVER\\_LIBRARY\\_SET](#) (SOLVER, SOLVER\_LIBRARY, ERR, ERROR,\*)

*Sets/changes the type of library to use for the solver.*

- subroutine **SOLVER\_ROUTINES::SOLVER\_LINEAR\_CREATE\_FINISH** (LINEAR\_SOLVER, ERR, ERROR,\*)

*Finishes the process of creating a linear solver.*

- subroutine **SOLVER\_ROUTINES::SOLVER\_LINEAR\_DIRECT\_CREATE\_FINISH** (LINEAR\_DIRECT\_SOLVER, ERR, ERROR,\*)

*Finishes the process of creating a linear direct solver.*

- subroutine **SOLVER\_ROUTINES::SOLVER\_LINEAR\_DIRECT\_FINALISE** (LINEAR\_DIRECT\_SOLVER, ERR, ERROR,\*)

*Finalise a direct linear solver for a linear solver and deallocate all memory.*

- subroutine **SOLVER\_ROUTINES::SOLVER\_LINEAR\_DIRECT\_INITIALISE** (LINEAR\_DIRECT\_SOLVER, ERR, ERROR,\*)

*Initialise a direct linear solver for a linear solver.*

- subroutine **SOLVER\_ROUTINES::SOLVER\_LINEAR\_DIRECT\_SOLVE** (LINEAR\_DIRECT\_SOLVER, ERR, ERROR,\*)

*Solve a linear direct solver.*

- subroutine **SOLVER\_ROUTINES::SOLVER\_LINEAR\_DIRECT\_TYPE\_SET** (SOLVER, DIRECT\_SOLVER\_TYPE, ERR, ERROR,\*)

*Sets/changes the type of direct linear solver.*

- subroutine **SOLVER\_ROUTINES::SOLVER\_LINEAR\_FINALISE** (LINEAR\_SOLVER, ERR, ERROR,\*)

*Finalise a linear solver for a problem solver.*

- subroutine **SOLVER\_ROUTINES::SOLVER\_LINEAR\_INITIALISE** (SOLVER, ERR, ERROR,\*)

*Initialise a linear solver for a problem solver.*

- subroutine **SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_ABSOLUTE\_TOLERANCE\_SET** (SOLVER, ABSOLUTE\_TOLERANCE, ERR, ERROR,\*)

*Sets/changes the maximum absolute tolerance for an iterative linear solver.*

- subroutine **SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_CREATE\_FINISH** (LINEAR\_ITERATIVE\_SOLVER, ERR, ERROR,\*)

*Finishes the process of creating a linear iterative solver.*

- subroutine **SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_DIVERGENCE\_TOLERANCE\_SET** (SOLVER, DIVERGENCE\_TOLERANCE, ERR, ERROR,\*)

*Sets/changes the maximum divergence tolerance for an iterative linear solver.*

- subroutine **SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_FINALISE** (LINEAR\_ITERATIVE\_SOLVER, ERR, ERROR,\*)

*Finalise an iterative linear solver for a linear solver and deallocate all memory.*

- subroutine **SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_INITIALISE** (LINEAR\_SOLVER, ERR, ERROR,\*)
 

*Initialise an iterative linear solver for a linear solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_MAXIMUM\_ITERATIONS\_SET** (SOLVER, MAXIMUM\_ITERATIONS, ERR, ERROR,\*)
 

*Sets/changes the maximum number of iterations for an iterative linear solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_PRECONDITIONER\_TYPE\_SET** (SOLVER, ITERATIVE\_PRECONDITIONER\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the type of preconditioner for an iterative linear solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_RELATIVE\_TOLERANCE\_SET** (SOLVER, RELATIVE\_TOLERANCE, ERR, ERROR,\*)
 

*Sets/changes the relative tolerance for an iterative linear solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_SOLVE** (LINEAR\_ITERATIVE\_SOLVER, ERR, ERROR,\*)
 

*Solves a linear iterative linear solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_TYPE\_SET** (SOLVER, ITERATIVE\_SOLVER\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the type of iterative linear solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_LINEAR\_SOLVE** (LINEAR\_SOLVER, ERR, ERROR,\*)
 

*Solve a linear solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_LINEAR\_TYPE\_SET** (SOLVER, LINEAR\_SOLVE\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the type of linear solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_MATRICES\_ASSEMBLE** (SOLVER, ERR, ERROR,\*)
 

*Assembles the solver matrices and rhs from the equations.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_ABSOLUTE\_TOLERANCE\_SET** (SOLVER, ABSOLUTE\_TOLERANCE, ERR, ERROR,\*)
 

*Sets/changes the maximum absolute tolerance for a nonlinear solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_CREATE\_FINISH** (NONLINEAR\_SOLVER, ERR, ERROR,\*)
 

*Finishes the process of creating a nonlinear solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_FINALISE** (NONLINEAR\_SOLVER, ERR, ERROR,\*)
 

*Finalise a nonlinear solver for a problem solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_INITIALISE** (SOLVER, ERR, ERROR,\*)
 

*Initialise a nonlinear solver for a problem solver.*

- subroutine **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_JACOBIAN\_EVALUATE** (SOLVER, ERR, ERROR,\*)
 

*Evaluates the Jacobian for a Newton like nonlinear solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_JACOBIAN\_EVALUATE\_PETSC** (SNES, X, A, B, FLAG, CTX)
 

*Called from the PETSc SNES solvers to evaluate the Jacobian for a Newton like nonlinear solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESearch\_ALPHA\_SET** (SOLVER, LINESEARCH\_ALPHA, ERR, ERROR,\*)
 

*Sets/changes the line search alpha for a nonlinear solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESearch\_CREATE\_FINISH** (NONLINEAR\_LINESearch\_SOLVER, ERR, ERROR,\*)
 

*Finishes the process of creating nonlinear Newton line search solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESearch\_FINALISE** (NONLINEAR\_LINESearch\_SOLVER, ERR, ERROR,\*)
 

*Finalise a nonlinear Newton line search solver and deallocate all memory.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESearch\_INITIALISE** (NONLINEAR\_SOLVER, ERR, ERROR,\*)
 

*Initialise a nonlinear Newton line search solver for a problem solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESearch\_MAXSTEP\_SET** (SOLVER, LINESEARCH\_MAXSTEP, ERR, ERROR,\*)
 

*Sets/changes the line search maximum step for a nonlinear solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESearch\_SOLVE** (NONLINEAR\_LINESearch\_SOLVER, ERR, ERROR,\*)
 • subroutine **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESearch\_STEPTOL\_SET** (SOLVER, LINESEARCH\_STEPTOL, ERR, ERROR,\*)
 

*Sets/changes the line search step tolerance for a nonlinear solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESearch\_TYPE\_SET** (SOLVER, LINESEARCH\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the line search type for a nonlinear solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_MAXIMUM\_FUNCTION\_EVALUATIONS\_SET** (SOLVER, MAXIMUM\_FUNCTION\_EVALUATIONS, ERR, ERROR,\*)
 

*Sets/changes the maximum number of function evaluations for a nonlinear solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_MAXIMUM\_ITERATIONS\_SET** (SOLVER, MAXIMUM\_ITERATIONS, ERR, ERROR,\*)
 

*Sets/changes the maximum number of iterations for a nonlinear solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_RELATIVE\_TOLERANCE\_SET** (SOLVER, RELATIVE\_TOLERANCE, ERR, ERROR,\*)
 

*Sets/changes the relative tolerance for a nonlinear solver.*

- subroutine `SOLVER_ROUTINES::SOLVER_NONLINEAR_RESIDUAL_EVALUATE` (SOLVER, ERR, ERROR,\*)
 

*Evaluates the residual for a Newton like nonlinear solver.*
- subroutine `SOLVER_ROUTINES::SOLVER_NONLINEAR_RESIDUAL_EVALUATE_PETSC` (SNES, X, F, CTX)
 

*Called from the PETSc SNES solvers to evaluate the residual for a Newton like nonlinear solver.*
- subroutine `SOLVER_ROUTINES::SOLVER_NONLINEAR SOLUTION_TOLERANCE_SET` (SOLVER, SOLUTION\_TOLERANCE, ERR, ERROR,\*)
 

*Sets/changes the solution tolerance for a nonlinear solver.*
- subroutine `SOLVER_ROUTINES::SOLVER_NONLINEAR_SOLVE` (NONLINEAR\_SOLVER, ERR, ERROR,\*)
 

*Sets/changes the trust region delta0 for a nonlinear solver.*
- subroutine `SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_CREATE_FINISH` (NONLINEAR\_TRUSTREGION\_SOLVER, ERR, ERROR,\*)
 

*Finishes the process of creating nonlinear trust region solver.*
- subroutine `SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_DELTA0_SET` (SOLVER, TRUSTREGION\_DELTA0, ERR, ERROR,\*)
 

*Sets/changes the trust region delta0 for a nonlinear solver.*
- subroutine `SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_FINALISE` (NONLINEAR\_TRUSTREGION\_SOLVER, ERR, ERROR,\*)
 

*Finalise a nonlinear trust region solver and deallocate all memory.*
- subroutine `SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_INITIALISE` (NONLINEAR\_SOLVER, ERR, ERROR,\*)
 

*Initialise a nonlinear trust region solver for a problem solver.*
- subroutine `SOLVER_ROUTINES::SOLVER_NONLINEAR_TRUSTREGION_solve` (NONLINEAR\_TRUSTREGION\_SOLVER, ERR, ERROR,\*)
 

*Sets/changes the trust region tolerance for a nonlinear solver.*
- subroutine `SOLVER_ROUTINES::SOLVER_NONLINEAR_TYPE_SET` (SOLVER, NONLINEAR\_SOLVE\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the type of nonlinear solver.*
- subroutine `SOLVER_ROUTINES::SOLVER_OUTPUT_TYPE_SET` (SOLVER, OUTPUT\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the output type for a solver.*
- subroutine `SOLVER_ROUTINES::SOLVER_SPARSITY_TYPE_SET` (SOLVER, SPARSITY\_TYPE, ERR, ERROR,\*)
 

*Sets/changes the sparsity type for a solver.*
- subroutine `SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_CREATE_FINISH` (TIME\_INTEGRATION\_SOLVER, ERR, ERROR,\*)
 

*Finishes the process of creating a time integration solver.*

- subroutine **SOLVER\_ROUTINES::SOLVER\_TIME\_INTEGRATION\_FINALISE** (TIME\_-INTEGRATION\_SOLVER, ERR, ERROR,\*)
 

*Finalise a time integration solver for a problem solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_TIME\_INTEGRATION\_INITIALISE** (SOLVER, ERR, ERROR,\*)
 

*Initialise a time integration solver for a problem solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_TIME\_INTEGRATION\_SOLVE** (TIME\_-INTEGRATION\_SOLVER, ERR, ERROR,\*)
 

*Solve a time integration solver.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_SOLVE** (SOLVER, ERR, ERROR,\*)
 

*Solve the problem.*
- subroutine **SOLVER\_ROUTINES::SOLVER\_VARIABLES\_UPDATE** (SOLVER, ERR, ERROR,\*)
 

*Updates the dependent variables from the solver solution.*

## Variables

- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_LINEAR\_TYPE** = 1
 

*Linear solution solver.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_TYPE** = 2
 

*A nonlinear solution solver.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_TIME\_INTEGRATION\_TYPE** = 3
 

*A time integration solver.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_EIGENPROBLEM\_TYPE** = 4
 

*A eigenproblem type.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_CMISS\_LIBRARY** = LIBRARY\_-CMISS\_TYPE
 

*CMISS (internal) solver library.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_PETSC\_LIBRARY** = LIBRARY\_-PETSC\_TYPE
 

*PETSc solver library.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_LINEAR\_DIRECT\_SOLVE\_-TYPE** = 1
 

*Direct linear solver type.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_LINEAR\_ITERATIVE\_SOLVE\_-TYPE** = 2
 

*Iterative linear solver type.*

*Iterative linear solver type.*

- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_DIRECT\_LU** = 1  
*LU direct linear solver.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_DIRECT\_CHOLESKY** = 2  
*Cholesky direct linear solver.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_DIRECT\_SVD** = 3  
*SVD direct linear solver.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_ITERATIVE\_RICHARDSON** = 1  
*Richardson iterative solver type.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_ITERATIVE\_CHEBYCHEV** = 2  
*Chebychev iterative solver type.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_ITERATIVE\_CONJUGATE\_GRADIENT** = 3  
*Conjugate gradient iterative solver type.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_ITERATIVE\_BICONJUGATE\_GRADIENT** = 4  
*Bi-conjugate gradient iterative solver type.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_ITERATIVE\_GMRES** = 5  
*Generalised minimum residual iterative solver type.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_ITERATIVE\_BICGSTAB** = 6  
*Stabilised bi-conjugate gradient iterative solver type.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_ITERATIVE\_CONJGRAD\_SQUARED** = 7  
*Conjugate gradient squared iterative solver type.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_ITERATIVE\_NO\_PRECONDITIONER** = 0  
*No preconditioner type.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_ITERATIVE\_JACOBI\_PRECONDITIONER** = 1  
*Jacobi preconditioner type.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_ITERATIVE\_BLOCK\_JACOBI\_PRECONDITIONER** = 2  
*Iterative block Jacobi preconditioner type.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_ITERATIVE\_SOR\_PRECONDITIONER** = 3

*Successive over relaxation preconditioner type.*

- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_ITERATIVE\_INCOMPLETE\_-CHOLESKY\_PRECONDITIONER = 4**

*Incomplete Cholesky preconditioner type.*

- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_ITERATIVE\_INCOMPLETE\_-LU\_PRECONDITIONER = 5**

*Incomplete LU preconditioner type.*

- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_ITERATIVE\_ADDITIVE\_-SCHWARZ\_PRECONDITIONER = 6**

*Additive Schwarz preconditioner type.*

- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_LINESEARCH = 1**

*Newton line search nonlinear solver type.*

- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_TRUSTREGION = 2**

*Newton trust region nonlinear solver type.*

- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_NONORMS\_-LINESEARCH = 1**

*No norms line search for Newton line search nonlinear solves.*

- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_NO\_-LINESEARCH = 2**

*No line search for Newton line search nonlinear solves.*

- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_QUADRATIC\_-LINESEARCH = 3**

*Quadratic search for Newton line search nonlinear solves.*

- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_CUBIC\_-LINESEARCH = 4**

*Cubic search for Newton line search nonlinear solves.*

- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_JACOBIAN\_-NOT\_CALCULATED = 1**

*The Jacobian values will not be calculated for the nonlinear equations set.*

- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_JACOBIAN\_-ANALYTIC\_CALCULATED = 2**

*The Jacobian values will be calculated analytically for the nonlinear equations set.*

- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_NONLINEAR\_JACOBIAN\_FD\_-CALCULATED = 3**

*The Jacobian values will be calculated using finite differences for the nonlinear equations set.*

- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_NO\_OUTPUT = 0**

*No output from the solver routines.*

- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_TIMING\_OUTPUT** = 1  
*Timing output from the solver routines.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_SOLVER\_OUTPUT** = 2  
*Timing and solver specific output from the solver routines.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_MATRIX\_OUTPUT** = 3  
*Timing and solver specific output and solution matrices output from the solver routines.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_SPARSE\_MATRICES** = 1  
*Use sparse solver matrices.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_FULL\_MATRICES** = 2  
*Use fully populated solver matrices.*
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_EULER\_INTEGRATOR** = 1
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_IMPROVED\_EULER\_INTEGRATOR** = 2
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_4TH\_RUNGE\_KUTTA\_INTEGRATOR** = 3
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_ADAMS\_MOULTON\_INTEGRATOR** = 4
- INTEGER(INTG), parameter **SOLVER\_ROUTINES::SOLVER\_LSODA\_INTEGRATOR** = 5

### 8.55.1 Detailed Description

#### Id

[solver\\_routines.f90](#) 181 2008-10-26 18:59:43Z chrisbradley

#### Author:

Chris Bradley This module handles all solver routines.

### 8.55.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [solver\\_routines.f90](#).

## 8.56 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/sorting.f90 File Reference

### Id

[sorting.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

### Classes

- interface [SORTING::BUBBLE\\_SORT](#)
- interface [SORTING::HEAP\\_SORT](#)
- interface [SORTING::SHELL\\_SORT](#)

### Namespaces

- namespace [SORTING](#)

*This module contains all procedures for sorting. NOTE: THE ROUTINES IN THIS MODULE HAVE NOT BEEN TESTED!!!*

### Functions

- subroutine [SORTING::BUBBLE\\_SORT\\_INTG](#) (A, ERR, ERROR,\*)
- subroutine [SORTING::BUBBLE\\_SORT\\_SP](#) (A, ERR, ERROR,\*)
- subroutine [SORTING::BUBBLE\\_SORT\\_DP](#) (A, ERR, ERROR,\*)
- subroutine [SORTING::HEAP\\_SORT\\_INTG](#) (A, ERR, ERROR,\*)
- subroutine [SORTING::HEAP\\_SORT\\_SP](#) (A, ERR, ERROR,\*)
- subroutine [SORTING::HEAP\\_SORT\\_DP](#) (A, ERR, ERROR,\*)
- subroutine [SORTING::SHELL\\_SORT\\_INTG](#) (A, ERR, ERROR,\*)
- subroutine [SORTING::SHELL\\_SORT\\_SP](#) (A, ERR, ERROR,\*)
- subroutine [SORTING::SHELL\\_SORT\\_DP](#) (A, ERR, ERROR,\*)

### 8.56.1 Detailed Description

#### Id

[sorting.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

#### Author:

Chris Bradley This module contains all procedures for sorting. NOTE: THE ROUTINES IN THIS MODULE HAVE NOT BEEN TESTED!!!

### 8.56.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [sorting.f90](#).

## 8.57 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/strings.f90 File Reference

### Id

[strings.f90](#) 177 2008-10-25 13:38:18Z chrispbradley

### Classes

- interface [STRINGS::IS\\_ABBREVIATION](#)  
*Returns .TRUE. if a supplied string is a valid abbreviation of a second supplied string.*
- interface [STRINGS::LIST\\_TO\\_CHARACTER](#)  
*Converts a list to its equivalent character string representation.*
- interface [STRINGS::NUMBER\\_TO\\_CHARACTER](#)  
*Converts a number to its equivalent character string representation.*
- interface [STRINGS::NUMBER\\_TO\\_VSTRING](#)  
*Converts a number to its equivalent varying string representation.*
- interface [STRINGS::STRING\\_TO\\_DOUBLE](#)  
*Converts a string representation of a number to a double precision number.*
- interface [STRINGS::STRING\\_TO\\_INTEGER](#)  
*Converts a string representation of a number to an integer.*
- interface [STRINGS::STRING\\_TO\\_LONG\\_INTEGER](#)  
*Converts a string representation of a number to a long integer.*
- interface [STRINGS::STRING\\_TO\\_LOGICAL](#)  
*Converts a string representation of a boolean value (TRUE or FALSE) to a logical.*
- interface [STRINGS::STRING\\_TO\\_SINGLE](#)  
*Converts a string representation of a number to a single precision number.*
- interface [STRINGS::CHARACTER\\_TO\\_LOWERCASE](#)  
*Returns a character string which is the lowercase equivalent of the supplied string.*
- interface [STRINGS::VSTRING\\_TO\\_LOWERCASE](#)  
*Returns a varying string which is the lowercase equivalent of the supplied string.*
- interface [STRINGS::CHARACTER\\_TO\\_UPPERCASE](#)  
*Returns a character string which is the uppercase equivalent of the supplied string.*
- interface [STRINGS::VSTRING\\_TO\\_UPPERCASE](#)  
*Returns a varying string which is the uppercase equivalent of the supplied string.*

## Namespaces

- namespace **STRINGS**

*This module contains all string manipulation and transformation routines.*

## Functions

- LOGICAL **STRINGS::IS\_ABBREVIATION\_C\_C** (SHORT, LONG, MIN\_NUM\_-CHARACTERS)

*IS\_ABBREVIATION* returns .TRUE. if the character string SHORT is an abbreviation of the character string LONG. SHORT must be at least MIN\_NUM\_CHARACTERS long.

- LOGICAL **STRINGS::IS\_ABBREVIATION\_C\_VS** (SHORT, LONG, MIN\_NUM\_-CHARACTERS)

*IS\_ABBREVIATION* returns .TRUE. if the character string SHORT is an abbreviation of the varying string LONG. SHORT must be at least MIN\_NUM\_CHARACTERS long.

- LOGICAL **STRINGS::IS\_ABBREVIATION\_VS\_C** (SHORT, LONG, MIN\_NUM\_-CHARACTERS)

*IS\_ABBREVIATION* returns .TRUE. if the varying string SHORT is an abbreviation of the character string LONG. SHORT must be at least MIN\_NUM\_CHARACTERS long.

- LOGICAL **STRINGS::IS\_ABBREVIATION\_VS\_VS** (SHORT, LONG, MIN\_NUM\_-CHARACTERS)

*IS\_ABBREVIATION* returns .TRUE. if the varying string SHORT is an abbreviation of the varying string LONG. SHORT must be at least MIN\_NUM\_CHARACTERS long.

- LOGICAL **STRINGS::IS\_DIGIT** (CHARAC)

*IS\_DIGIT* returns .TRUE. if the character CHARAC is a digit character (i.e. 0..9).

- LOGICAL **STRINGS::IS\_LETTER** (CHARAC)

*IS\_LETTER* returns .TRUE. if the character CHARAC is a letter character (i.e. A..Z or a..z).

- LOGICAL **STRINGS::IS\_LOWERCASE** (CHARC)

*Returns* .TRUE. if the supplied character is a lowercase character.

- LOGICAL **STRINGS::IS\_UPPERCASE** (CHARC)

*Returns* .TRUE. if the supplied character is an uppercase character.

- LOGICAL **STRINGS::IS\_WHITESPACE** (CHARAC)

*IS\_WHITESPACE* returns .TRUE. if the character CHARAC is a whitespace character (i.e. space, tabs, etc.).

- CHARACTER(LEN=MAXSTRLEN) **STRINGS::LIST\_TO\_CHARACTER\_C** (NUMBER\_IN\_-LIST, LIST, FORMAT, ERR, ERROR, LIST\_LENGTHS)

*Converts a character list to its equivalent character string representation as determined by the supplied format. If present, the optional argument LIST\_LENGTHS is used for the lengths of each list elements length otherwise the trimmed length is used. NOTE: The FORMAT is ignored for this child FUNCTION.*

- CHARACTER(LEN=MAXSTRLEN) [STRINGS::LIST\\_TO\\_CHARACTER\\_INTG](#) (NUMBER\_IN\_LIST, LIST, FORMAT, ERR, ERROR)
 

*Converts an integer list to its equivalent character string representation as determined by the supplied format.*
- CHARACTER(LEN=MAXSTRLEN) [STRINGS::LIST\\_TO\\_CHARACTER\\_LINTG](#) (NUMBER\_IN\_LIST, LIST, FORMAT, ERR, ERROR)
 

*Converts an long integer list to its equivalent character string representation as determined by the supplied format.*
- CHARACTER(LEN=MAXSTRLEN) [STRINGS::LIST\\_TO\\_CHARACTER\\_L](#) (NUMBER\_IN\_LIST, LIST, FORMAT, ERR, ERROR)
 

*Converts a logical list to its equivalent character string representation as determined by the supplied format string. The FORMAT is ignored for this child FUNCTION.*
- CHARACTER(LEN=MAXSTRLEN) [STRINGS::LIST\\_TO\\_CHARACTER\\_SP](#) (NUMBER\_IN\_LIST, LIST, FORMAT, ERR, ERROR)
 

*Converts a single precision list to its equivalent character string representation as determined by the supplied format string.*
- CHARACTER(LEN=MAXSTRLEN) [STRINGS::LIST\\_TO\\_CHARACTER\\_DP](#) (NUMBER\_IN\_LIST, LIST, FORMAT, ERR, ERROR)
 

*Converts a double precision list to its equivalent character string representation as determined by the supplied format string.*
- CHARACTER(LEN=MAXSTRLEN) [STRINGS::LOGICAL\\_TO\\_CHARACTER](#) (LOGICAL\_VALUE, ERR, ERROR)
 

*Converts a logical value to either a "TRUE" or "FALSE" character string.*
- TYPE(VARYING\_STRING) [STRINGS::LOGICAL\\_TO\\_VSTRING](#) (LOGICALVALUE, ERR, ERROR)
 

*Converts a logical value to either a "TRUE" or "FALSE" varying string.*
- CHARACTER(LEN=MAXSTRLEN) [STRINGS::NUMBER\\_TO\\_CHARACTER\\_INTG](#) (NUMBER, FORMAT, ERR, ERROR)
 

*Converts an integer number to its equivalent character string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g. "I3".*
- CHARACTER(LEN=MAXSTRLEN) [STRINGS::NUMBER\\_TO\\_CHARACTER\\_LINTG](#) (NUMBER, FORMAT, ERR, ERROR)
 

*Converts a long integer number to its equivalent character string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g. "I3".*
- CHARACTER(LEN=MAXSTRLEN) [STRINGS::NUMBER\\_TO\\_CHARACTER\\_SP](#) (NUMBER, FORMAT, ERR, ERROR)
 

*Converts a single precision number to its equivalent character string representation as determined by the supplied format string. NOTE: If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="\*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.*
- CHARACTER(LEN=MAXSTRLEN) [STRINGS::NUMBER\\_TO\\_CHARACTER\\_DP](#) (NUMBER, FORMAT, ERR, ERROR)
 

*Converts a double precision number to its equivalent character string representation as determined by the supplied format string. NOTE: If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="\*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.*

*Converts a double precision number to its equivalent character string representation as determined by the supplied format string. Note If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="\*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.*

- TYPE(VARYING\_STRING) [STRINGS::NUMBER\\_TO\\_VSTRING\\_INTG](#) (NUMBER, FORMAT, ERR, ERROR)

*Converts an integer number to its equivalent varying string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g. "I3".*

- TYPE(VARYING\_STRING) [STRINGS::NUMBER\\_TO\\_VSTRING\\_LINTG](#) (NUMBER, FORMAT, ERR, ERROR)

*Converts a long integer number to its equivalent varying string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g., "I3".*

- TYPE(VARYING\_STRING) [STRINGS::NUMBER\\_TO\\_VSTRING\\_SP](#) (NUMBER, FORMAT, ERR, ERROR)

*Converts a single precision number to its equivalent varying string representation as determined by the supplied format string. Note If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="\*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.*

- TYPE(VARYING\_STRING) [STRINGS::NUMBER\\_TO\\_VSTRING\\_DP](#) (NUMBER, FORMAT, ERR, ERROR)

*Converts a double precision number to its equivalent varying string representation as determined by the supplied format string. Note If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="\*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.*

- REAL(DP) [STRINGS::STRING\\_TO\\_DOUBLE\\_C](#) (STRING, ERR, ERROR)

*Converts a character string representation of a number to a double precision number.*

- REAL(DP) [STRINGS::STRING\\_TO\\_DOUBLE\\_VS](#) (STRING, ERR, ERROR)

*Converts a varying string representation of a number to a double precision number.*

- INTEGER(INTG) [STRINGS::STRING\\_TO\\_INTEGER\\_C](#) (STRING, ERR, ERROR)

*Converts a character string representation of a number to an integer.*

- INTEGER(INTG) [STRINGS::STRING\\_TO\\_INTEGER\\_VS](#) (STRING, ERR, ERROR)

*Converts a varying string representation of a number to an integer.*

- INTEGER(LINTG) [STRINGS::STRING\\_TO\\_LONG\\_INTEGER\\_C](#) (STRING, ERR, ERROR)

*Converts a character string representation of a number to a long integer.*

- INTEGER(LINTG) [STRINGS::STRING\\_TO\\_LONG\\_INTEGER\\_VS](#) (STRING, ERR, ERROR)

*Converts a varying string representation of a number to a long integer.*

- LOGICAL [STRINGS::STRING\\_TO\\_LOGICAL\\_C](#) (STRING, ERR, ERROR)

*Converts a character string representation of a boolean (TRUE or FALSE) to a logical.*

- LOGICAL [STRINGS::STRING\\_TO\\_LOGICAL\\_VS](#) (STRING, ERR, ERROR)

*Converts a varying string representation of a boolean (TRUE or FALSE) to a logical.*

- REAL(SP) **STRINGS::STRING\_TO\_SINGLE\_C** (STRING, ERR, ERROR)  
*Converts a character string representation of a number to a single precision number.*
- REAL(SP) **STRINGS::STRING\_TO\_SINGLE\_VS** (STRING, ERR, ERROR)  
*Converts a varying string representation of a number to a single precision number.*
- function **STRINGS::CHARACTER\_TO\_LOWERCASE\_C** (STRING)  
• function **STRINGS::CHARACTER\_TO\_LOWERCASE\_VS** (STRING)  
*Returns a character string that is the lowercase equivalent of the supplied varying string.*
- TYPE(VARYING\_STRING) **STRINGS::VSTRING\_TO\_LOWERCASE\_C** (STRING)  
*Returns a varying string that is the lowercase equivalent of the supplied character string.*
- TYPE(VARYING\_STRING) **STRINGS::VSTRING\_TO\_LOWERCASE\_VS** (STRING)  
*Returns a varying string that is the lowercase equivalent of the supplied varying string.*
- function **STRINGS::CHARACTER\_TO\_UPPERCASE\_C** (STRING)  
*Returns a character string which is uppercase equivalent of the supplied character string.*
- function **STRINGS::CHARACTER\_TO\_UPPERCASE\_VS** (STRING)  
*Returns a character string which is uppercase equivalent of the supplied varying string.*
- TYPE(VARYING\_STRING) **STRINGS::VSTRING\_TO\_UPPERCASE\_C** (STRING)  
*Returns a varying string which is uppercase equivalent of the supplied character string.*
- TYPE(VARYING\_STRING) **STRINGS::VSTRING\_TO\_UPPERCASE\_VS** (STRING)  
*Returns a varying string which is uppercase equivalent of the supplied varying string.*

### 8.57.1 Detailed Description

#### Id

[strings.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

#### Author:

Chris Bradley This module contains all string manipulation and transformation routines.

### 8.57.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [strings.f90](#).

## 8.58 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/timer\_c.c File Reference

Include dependency graph for timer\_c.c:

### Defines

- #define **USER\_CPU** 1
- #define **SYSTEM\_CPU** 2
- #define **TOTAL\_CPU** 3

### Functions

- void **CPUTimer** (double \*return\_time, int \*flag, int \*err, char \*error\_string)

#### 8.58.1 Define Documentation

##### 8.58.1.1 #define SYSTEM\_CPU 2

Referenced by CPUTimer().

##### 8.58.1.2 #define TOTAL\_CPU 3

Referenced by CPUTimer().

##### 8.58.1.3 #define USER\_CPU 1

Referenced by CPUTimer().

#### 8.58.2 Function Documentation

##### 8.58.2.1 void **CPUTimer** (*double \*return\_time, int \*flag, int \*err, char \*error\_string*)

Definition at line 107 of file timer\_c.c.

References SYSTEM\_CPU, TOTAL\_CPU, and USER\_CPU.

## 8.59 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/timer\_f.f90 File Reference

### Id

[timer\\_f.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

### Classes

- interface [TIMER::interface](#)

### Namespaces

- namespace [TIMER](#)

*This module contains routines for timing the program.*

### Functions

- subroutine [TIMER::CPU\\_TIMER](#) (TIME\_TYPE, TIME, ERR, ERROR,\*)

### Variables

- INTEGER(INTG), parameter [TIMER::USER\\_CPU](#) = 1
- INTEGER(INTG), parameter [TIMER::SYSTEM\\_CPU](#) = 2
- INTEGER(INTG), parameter [TIMER::TOTAL\\_CPU](#) = 3

### 8.59.1 Detailed Description

#### Id

[timer\\_f.f90](#) 177 2008-10-25 13:38:18Z chrisbradley

#### Author:

Chris Bradley This module contains routines for timing the program.

### 8.59.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University

of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [timer\\_f.f90](#).

## 8.60 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/trees.f90 File Reference

### Id

[trees.f90](#) 178 2008-10-26 13:07:57Z chrisbradley

### Classes

- struct [TREES::TREE\\_NODE\\_TYPE](#)
- struct [TREES::TREE\\_TYPE](#)

### Namespaces

- namespace [TREES](#)  
*Implements trees of base types.*

### Functions

- subroutine [TREES::TREE\\_CREATE\\_FINISH](#) (TREE, ERR, ERROR,\*)  
*Finishes the creation of a tree created with TREE\_CREATE\_START.*
- subroutine [TREES::TREE\\_CREATE\\_START](#) (TREE, ERR, ERROR,\*)  
*Starts the creation of a tree and returns a pointer to the created tree.*
- subroutine [TREES::TREE\\_DESTROY](#) (TREE, ERR, ERROR,\*)  
*Destroys a tree.*
- subroutine [TREES::TREE\\_DETACH\\_AND\\_DESTROY](#) (TREE, NUMBER\_IN\_TREE, TREE\_VALUES, ERR, ERROR,\*)  
*Detaches the tree values and returns them as a pointer to the an array and then destroys the tree.*
- subroutine [TREES::TREE\\_DETACH\\_IN\\_ORDER](#) (TREE, X, COUNT, TREE\_VALUES, ERR, ERROR,\*)  
*Detaches the tree values in order from the specified tree node and adds them to the tree values array.*
- subroutine [TREES::TREE\\_FINALISE](#) (TREE, ERR, ERROR,\*)  
*Finalises a tree and deallocates all memory.*
- subroutine [TREES::TREE\\_INITIALISE](#) (TREE, ERR, ERROR,\*)  
*Initialises a tree.*
- subroutine [TREES::TREE\\_INSERT\\_TYPE\\_SET](#) (TREE, INSERT\_TYPE, ERR, ERROR,\*)  
*Sets/changes the insert type for a tree.*
- subroutine [TREES::TREE\\_ITEM\\_DELETE](#) (TREE, KEY, ERR, ERROR,\*)  
*Deletes a tree node specified by a key from a tree.*

- subroutine [TREES::TREE\\_ITEM\\_INSERT](#) (TREE, KEY, VALUE, INSERT\_STATUS, ERR, ERROR,\*)
 

*Inserts a tree node into a red-black tree.*
- subroutine [TREES::TREE\\_NODE\\_FINALISE](#) (TREE, TREE\_NODE, ERR, ERROR,\*)
 

*Finalises a tree node and deallocates all memory.*
- subroutine [TREES::TREE\\_NODE\\_INITIALISE](#) (TREE, TREE\_NODE, ERR, ERROR,\*)
 

*Initialises a tree node.*
- subroutine [TREES::TREE\\_NODE\\_KEY\\_GET](#) (TREE, TREE\_NODE, KEY, ERR, ERROR,\*)
 

*Gets the key at a specified tree node.*
- subroutine [TREES::TREE\\_NODE\\_VALUE\\_GET](#) (TREE, TREE\_NODE, VALUE, ERR, ERROR,\*)
 

*Gets the value at a specified tree node.*
- subroutine [TREES::TREE\\_NODE\\_VALUE\\_SET](#) (TREE, TREE\_NODE, VALUE, ERR, ERROR,\*)
 

*Sets the value at a specified tree node.*
- subroutine [TREES::TREE\\_OUTPUT](#) (ID, TREE, ERR, ERROR,\*)
 

*Outputs a tree to the specified output stream ID.*
- subroutine [TREES::TREE\\_OUTPUT\\_IN\\_ORDER](#) (ID, TREE, X, ERR, ERROR,\*)
 

*Outputs a tree in order to the specified output stream ID from the specified tree node.*
- TYPE(TREE\_NODE\_TYPE), pointer [TREES::TREE\\_PREDECESSOR](#) (TREE, X, ERR, ERROR)
 

*Returns the predecessor of a tree at a specified tree node.*
- subroutine [TREES::TREE\\_SEARCH](#) (TREE, KEY, X, ERR, ERROR,\*)
 

*Searches a tree to see if it contains a key.*
- TYPE(TREE\_NODE\_TYPE), pointer [TREES::TREE\\_SUCCESSOR](#) (TREE, X, ERR, ERROR)
 

*Returns the successor of a tree at a specified tree node.*

## Variables

- INTEGER(INTG), parameter [TREES::TREE\\_BLACK\\_NODE](#) = 0
 

*The black colour type for a tree node.*
- INTEGER(INTG), parameter [TREES::TREE\\_RED\\_NODE](#) = 1
 

*The red colour type for a tree node.*
- INTEGER(INTG), parameter [TREES::TREE\\_NODE\\_INSERT\\_SUCESSFUL](#) = 1
 

*Successful insert status.*
- INTEGER(INTG), parameter [TREES::TREE\\_NODE\\_DUPLICATE\\_KEY](#) = 2

*Duplicate key found for those trees that do not allow duplicate keys.*

- INTEGER(INTG), parameter [TREES::TREE\\_DUPLICATES\\_ALLOWED\\_TYPE](#) = 1  
*Duplicate keys allowed tree type.*

- INTEGER(INTG), parameter [TREES::TREE\\_NO\\_DUPLICATES\\_ALLOWED](#) = 2  
*No duplicate keys allowed tree type.*

### 8.60.1 Detailed Description

#### Id

[trees.f90](#) 178 2008-10-26 13:07:57Z chrispb

#### Author:

Chris Bradley Implements trees of base types.

### 8.60.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

#### Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [trees.f90](#).

## 8.61 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/types.f90 File Reference

### Id

[types.f90](#) 181 2008-10-26 18:59:43Z chrispb

### Classes

- struct [TYPES::QUADRATURE\\_SCHEME\\_TYPE](#)  
*Contains information for a particular quadrature scheme.*
- struct [TYPES::QUADRATURE\\_SCHEME\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [QUADRATURE\\_SCHEME\\_TYPE](#).*
- struct [TYPES::QUADRATURE\\_TYPE](#)  
*Contains information on the quadrature to be used for integrating a basis.*
- struct [TYPES::BASIS\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [BASIS\\_TYPE](#).*
- struct [TYPES::BASIS\\_TYPE](#)  
*Contains all information about a basis .*
- struct [TYPES::BASIS\\_FUNCTIONS\\_TYPE](#)  
*Contains information on the defined basis functions.*
- struct [TYPES::COORDINATE\\_SYSTEM\\_TYPE](#)  
*Contains information on a coordinate system.*
- struct [TYPES::NODE\\_TYPE](#)  
*Contains information about a node.*
- struct [TYPES::NODES\\_TYPE](#)  
*Contains information on the nodes defined on a region.*
- struct [TYPES::MESH\\_DOFS\\_TYPE](#)  
*Contains information on the dofs for a mesh.*
- struct [TYPES::MESH\\_ELEMENT\\_TYPE](#)  
*Contains the information for an element in a mesh.*
- struct [TYPES::MESH\\_ELEMENTS\\_TYPE](#)  
*Contains the information for the elements of a mesh.*
- struct [TYPES::MESH\\_NODE\\_TYPE](#)  
*Contains the topology information for a global node of a mesh.*
- struct [TYPES::MESH\\_NODES\\_TYPE](#)  
*Contains the information for the nodes of a mesh.*

- struct [TYPES::MESH\\_TOPOLOGY\\_TYPE](#)  
*Contains information on the (global) topology of a mesh.*
- struct [TYPES::MESH\\_TOPOLOGY\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [MESH\\_TOPOLOGY\\_TYPE](#).*
- struct [TYPES::MESH\\_TYPE](#)  
*Contains information on a mesh defined on a region.*
- struct [TYPES::MESH\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [MESH\\_TYPE](#).*
- struct [TYPES::MESHES\\_TYPE](#)  
*Contains information on the meshes defined on a region.*
- struct [TYPES::GENERATED\\_MESH\\_REGULAR\\_TYPE](#)  
*Contains information on a generated regular mesh.*
- struct [TYPES::GENERATED\\_MESH\\_TYPE](#)  
*Contains information on a generated mesh.*
- struct [TYPES::GENERATED\\_MESH\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [GENERATED\\_MESH\\_TYPE](#).*
- struct [TYPES::GENERATED\\_MESHES\\_TYPE](#)  
*Contains information on the generated meshes defined on a region.*
- struct [TYPES::DOMAIN\\_DOFS\\_TYPE](#)  
*Contains information on the degrees-of-freedom (dofs) for a domain.*
- struct [TYPES::DOMAIN\\_LINE\\_TYPE](#)  
*Contains the information for a line in a domain.*
- struct [TYPES::DOMAIN\\_LINE\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [DOMAIN\\_LINE\\_TYPE](#).*
- struct [TYPES::DOMAIN\\_LINES\\_TYPE](#)  
*Contains the topology information for the lines of a domain.*
- struct [TYPES::DOMAIN\\_FACE\\_TYPE](#)  
*Contains the information for a face in a domain.*
- struct [TYPES::DOMAIN\\_FACE\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [FIELD\\_VARIABLE\\_TYPE](#).*
- struct [TYPES::DOMAIN\\_FACES\\_TYPE](#)  
*Contains the topology information for the faces of a domain.*
- struct [TYPES::DOMAIN\\_ELEMENT\\_TYPE](#)

*Contains the information for an element in a domain.*

- struct [TYPES::DOMAIN\\_ELEMENTS\\_TYPE](#)  
*Contains the topology information for the elements of a domain.*
- struct [TYPES::DOMAIN\\_NODE\\_TYPE](#)  
*Contains the topology information for a local node of a domain.*
- struct [TYPES::DOMAIN\\_NODES\\_TYPE](#)  
*Contains the topology information for the nodes of a domain.*
- struct [TYPES::DOMAIN\\_TOPOLOGY\\_TYPE](#)  
*Contains the topology information for a domain.*
- struct [TYPES::DISTRIBUTED\\_VECTOR\\_TRANSFER\\_TYPE](#)  
*Contains the information for an adjacent domain for transferring the ghost data of a distributed vector to/from the current domain.*
- struct [TYPES::DISTRIBUTED\\_VECTOR\\_CMISS\\_TYPE](#)  
*Contains information for a CMISS distributed vector.*
- struct [TYPES::DISTRIBUTED\\_VECTOR\\_PETSC\\_TYPE](#)  
*Contains information for a PETSc distributed vector.*
- struct [TYPES::DISTRIBUTED\\_VECTOR\\_TYPE](#)  
*Contains the information for a vector that is distributed across a number of domains.*
- struct [TYPES::DISTRIBUTED\\_MATRIX\\_CMISS\\_TYPE](#)  
*Contains information for a CMISS distributed matrix.*
- struct [TYPES::DISTRIBUTED\\_MATRIX\\_PETSC\\_TYPE](#)  
*Contains information for a PETSc distributed matrix.*
- struct [TYPES::DISTRIBUTED\\_MATRIX\\_TYPE](#)  
*Contains the information for a matrix that is distributed across a number of domains.*
- struct [TYPES::VECTOR\\_TYPE](#)  
*Contains information for a vector.*
- struct [TYPES::MATRIX\\_TYPE](#)  
*Contains information for a matrix.*
- struct [TYPES::DOMAIN\\_ADJACENT\\_DOMAIN\\_TYPE](#)  
*Contains the information on an adjacent domain to a domain in a domain mapping.*
- struct [TYPES::DOMAIN\\_GLOBAL\\_MAPPING\\_TYPE](#)  
*Contains the local information for a global mapping number for a domain mapping.*
- struct [TYPES::DOMAIN\\_MAPPING\\_TYPE](#)  
*Contains information on the domain mappings (i.e., local and global numberings).*

- struct [TYPES::DOMAIN\\_MAPPINGS\\_TYPE](#)  
*Contains information on the domain decomposition mappings.*
- struct [TYPES::DOMAIN\\_TYPE](#)  
*A pointer to the domain decomposition for this domain.*
- struct [TYPES::DOMAIN\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [DOMAIN\\_TYPE](#).*
- struct [TYPES::DECOMPOSITION\\_LINE\\_TYPE](#)  
*Contains the information for a line in a decomposition.*
- struct [TYPES::DECOMPOSITION\\_LINES\\_TYPE](#)  
*Contains the topology information for the lines of a decomposition.*
- struct [TYPES::DECOMPOSITION\\_ELEMENT\\_TYPE](#)  
*Contains the information for an element in a decomposition.*
- struct [TYPES::DECOMPOSITION\\_ELEMENTS\\_TYPE](#)  
*Contains the topology information for the elements of a decomposition.*
- struct [TYPES::DECOMPOSITION\\_TOPOLOGY\\_TYPE](#)  
*Contains the topology information for a decomposition.*
- struct [TYPES::DECOMPOSITION\\_TYPE](#)  
*Contains information on the domain decomposition.*
- struct [TYPES::DECOMPOSITION\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [DECOMPOSITION\\_TYPE](#).*
- struct [TYPES::DECOMPOSITIONS\\_TYPE](#)  
*Contains information on the domain decompositions defined on a mesh.*
- struct [TYPES::FIELD\\_INTERPOLATED\\_POINT\\_METRICS\\_TYPE](#)  
*Contains the interpolated point coordinate metrics. Old [CMISS](#) name GL, GU, RG.*
- struct [TYPES::FIELD\\_INTERPOLATED\\_POINT\\_TYPE](#)  
*Contains the interpolated value (and the derivatives wrt xi) of a field at a point. Old [CMISS](#) name XG.*
- struct [TYPES::FIELD\\_INTERPOLATION\\_PARAMETERS\\_TYPE](#)  
*Contains the parameters required to interpolate a field variable within an element. Old [CMISS](#) name XE.*
- struct [TYPES::FIELD\\_GEOMETRIC\\_PARAMETERS\\_TYPE](#)  
*Contains the geometric parameters (lines, faces, volumes etc.) for a geometric field decomposition.*
- struct [TYPES::FIELD\\_SCALING\\_TYPE](#)  
*A type to hold the scale factors for the appropriate mesh component of a field.*
- struct [TYPES::FIELD\\_SCALINGS\\_TYPE](#)

A type to hold the field scalings for the field.

- struct [TYPES::FIELD\\_DOF\\_TO\\_PARAM\\_MAP\\_TYPE](#)  
*A type to hold the mapping from field dof numbers to field parameters (nodes, elements, etc).*
- struct [TYPES::FIELD\\_PARAM\\_TO\\_DOF\\_MAP\\_TYPE](#)  
*A type to hold the mapping from field parameters (nodes, elements, etc) to field dof numbers for a particular field variable component.*
- struct [TYPES::FIELD\\_VARIABLE\\_COMPONENT\\_TYPE](#)  
*Contains information for a component of a field variable.*
- struct [TYPES::FIELD\\_VARIABLE\\_TYPE](#)  
*Contains information for a field variable defined on a field.*
- struct [TYPES::FIELD\\_VARIABLE\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [FIELD\\_VARIABLE\\_TYPE](#).*
- struct [TYPES::FIELD\\_CREATE\\_VALUES\\_CACHE\\_TYPE](#)  
*A type to temporarily hold (cache) the user modifiable values which are used to create a field.*
- struct [TYPES::FIELD\\_MAPPINGS\\_TYPE](#)  
*The type containing the mappings for the field.*
- struct [TYPES::FIELD\\_PARAMETER\\_SET\\_TYPE](#)  
*A type to hold the parameter sets for a field.*
- struct [TYPES::FIELD\\_PARAMETER\\_SET\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [FIELD\\_PARAMETER\\_SET\\_TYPE](#).*
- struct [TYPES::FIELD\\_PARAMETER\\_SETS\\_TYPE](#)  
*A type to store the parameter sets for a field.*
- struct [TYPES::FIELD\\_TYPE](#)  
*Contains information for a field defined on a region.*
- struct [TYPES::FIELD\\_PTR\\_TYPE](#)  
*A buffer type to allow for an array of pointers to a [FIELD\\_TYPE](#).*
- struct [TYPES::FIELDS\\_TYPE](#)  
*Contains information on the fields defined on a region.*
- struct [TYPES::ELEMENT\\_MATRIX\\_TYPE](#)  
*Contains information for an element matrix.*
- struct [TYPES::ELEMENT\\_VECTOR\\_TYPE](#)  
*Contains information for an element vector.*
- struct [TYPES::EQUATIONS\\_MATRIX\\_TYPE](#)  
*Contains information about an equations matrix.*

- struct [TYPES::EQUATIONS\\_MATRIX\\_PTR\\_TYPE](#)
- struct [TYPES::EQUATIONS\\_JACOBIAN\\_TYPE](#)

*Contains information on the Jacobian matrix for nonlinear problems.*

- struct [TYPES::EQUATIONS\\_MATRICES\\_LINEAR\\_TYPE](#)
- struct [TYPES::EQUATIONS\\_MATRICES\\_NONLINEAR\\_TYPE](#)
- struct [TYPES::EQUATIONS\\_MATRICES\\_RHS\\_TYPE](#)
- struct [TYPES::EQUATIONS\\_MATRICES\\_SOURCE\\_TYPE](#)
- struct [TYPES::EQUATIONS\\_MATRICES\\_TYPE](#)

*Contains information on the equations matrices and rhs vector.*

- struct [TYPES::VARIABLE\\_TO\\_EQUATIONS\\_COLUMN\\_MAP\\_TYPE](#)

*Contains the information about the mapping of a variable DOF to an equations matrix column.*

- struct [TYPES::VARIABLE\\_TO\\_EQUATIONS\\_MATRICES\\_MAP\\_TYPE](#)

*Contains the mapping for a dependent variable type to the equations matrices.*

- struct [TYPES::EQUATIONS\\_MATRIX\\_TO\\_VARIABLE\\_MAP\\_TYPE](#)
- struct [TYPES::EQUATIONS\\_MAPPING\\_LINEAR\\_TYPE](#)
- struct [TYPES::EQUATIONS\\_JACOBIAN\\_TO\\_VARIABLE\\_MAP\\_TYPE](#)
- struct [TYPES::VARIABLE\\_TO\\_EQUATIONS\\_JACOBIAN\\_MAP\\_TYPE](#)

*Contains the mapping for a dependent variable type to the nonlinear Jacobian matrix.*

- struct [TYPES::EQUATIONS\\_MAPPING\\_NONLINEAR\\_TYPE](#)
- struct [TYPES::EQUATIONS\\_MAPPING\\_RHS\\_TYPE](#)
- struct [TYPES::EQUATIONS\\_MAPPING\\_SOURCE\\_TYPE](#)
- struct [TYPES::EQUATIONS\\_MAPPING\\_CREATE\\_VALUES\\_CACHE\\_TYPE](#)
- struct [TYPES::EQUATIONS\\_MAPPING\\_TYPE](#)
- struct [TYPES::EQUATIONS\\_INTERPOLATION\\_TYPE](#)

*Contains information on the interpolation for the equations.*

- struct [TYPES::EQUATIONS\\_LINEAR\\_DATA\\_TYPE](#)

*Contains information on any data required for a linear solution.*

- struct [TYPES::EQUATIONS\\_NONLINEAR\\_DATA\\_TYPE](#)

*Contains information on any data required for a non-linear solution.*

- struct [TYPES::EQUATIONS\\_TIME\\_DATA\\_TYPE](#)

*Contains information on any data required for a time-dependent equations.*

- struct [TYPES::EQUATIONS\\_TYPE](#)

*Contains information about the equations in an equations set.*

- struct [TYPES::EQUATIONS\\_SET\\_GEOMETRY\\_TYPE](#)

*Contains information on the geometry for an equations set.*

- struct [TYPES::EQUATIONS\\_SET\\_MATERIALS\\_TYPE](#)

- struct [TYPES::EQUATIONS\\_SET\\_FIXED\\_CONDITIONS\\_TYPE](#)

*Contains information on the fixed conditions for the problem.*

- struct [TYPES::EQUATIONS\\_SET\\_DEPENDENT\\_TYPE](#)  
*Contains information on the dependent variables for the equations set.*
- struct [TYPES::EQUATIONS\\_SET\\_SOURCE\\_TYPE](#)  
*Contains information on the source for the equations set.*
- struct [TYPES::EQUATIONS\\_SET\\_ANALYTIC\\_TYPE](#)  
*Contains information on the analytic setup for the equations set.*
- struct [TYPES::EQUATIONS\\_SET\\_TYPE](#)  
*Contains information on an equations set.*
- struct [TYPES::EQUATIONS\\_SET\\_PTR\\_TYPE](#)
- struct [TYPES::EQUATIONS\\_SETS\\_TYPE](#)
- struct [TYPES::SOLVER\\_MATRIX\\_TYPE](#)  
*Contains information on the solver matrix.*
- struct [TYPES::SOLVER\\_MATRIX\\_PTR\\_TYPE](#)
- struct [TYPES::SOLVER\\_JACOBIAN\\_TYPE](#)  
*Contains information on the solver Jacobian for nonlinear problems.*
- struct [TYPES::SOLVER\\_MATRICES\\_TYPE](#)  
*Contains information on the solver matrices and rhs vector.*
- struct [TYPES::LINEAR\\_DIRECT\\_SOLVER\\_TYPE](#)  
*Contains information for a direct linear solver.*
- struct [TYPES::LINEAR\\_ITERATIVE\\_SOLVER\\_TYPE](#)  
*Contains information for a direct linear solver.*
- struct [TYPES::LINEAR\\_SOLVER\\_TYPE](#)  
*Contains information for a linear solver.*
- struct [TYPES::NONLINEAR\\_LINESearch\\_SOLVER\\_TYPE](#)  
*Contains information for a Newton line search nonlinear solver.*
- struct [TYPES::NONLINEAR\\_TRUSTREGION\\_SOLVER\\_TYPE](#)  
*Contains information for a Newton trust region nonlinear solver.*
- struct [TYPES::NONLINEAR\\_SOLVER\\_TYPE](#)  
*Contains information for a nonlinear solver.*
- struct [TYPES::TIME\\_INTEGRATION\\_SOLVER\\_TYPE](#)  
*Contains information for a time integration solver.*
- struct [TYPES::EIGENPROBLEM\\_SOLVER\\_TYPE](#)  
*Contains information for a time integration solver.*
- struct [TYPES::SOLVER\\_TYPE](#)

*Contains information on the type of solver to be used.*

- struct [TYPES::EQUATIONS\\_COL\\_TO\\_SOLVER\\_COLS\\_MAP\\_TYPE](#)
- struct [TYPES::EQUATIONS\\_TO\\_SOLVER\\_MAPS\\_TYPE](#)
- struct [TYPES::EQUATIONS\\_TO\\_SOLVER\\_MAPS\\_PTR\\_TYPE](#)

*A buffer type to allow for an array of pointers to a [EQUATIONS\\_TO\\_SOLVER\\_MAPS\\_TYPE](#).*

- struct [TYPES::JACOBIAN\\_COL\\_TO\\_SOLVER\\_COLS\\_MAP\\_TYPE](#)
- struct [TYPES::JACOBIAN\\_TO\\_SOLVER\\_MAP\\_TYPE](#)
- struct [TYPES::VARIABLE\\_TO\\_SOLVER\\_COL\\_MAP\\_TYPE](#)

*Contains information on the mappings between field variable dofs in an equation set and the solver matrix columns (solver dofs).*

- struct [TYPES::EQUATIONS\\_TO\\_SOLVER\\_MATRIX\\_MAPS\\_SM\\_TYPE](#)

*Contains information on the equations to solver matrix mappings when indexing by solver matrix number.*

- struct [TYPES::EQUATIONS\\_TO\\_SOLVER\\_MATRIX\\_MAPS\\_EM\\_TYPE](#)

*Contains information on the equations to solver matrix mappings when indexing by equations matrix number.*

- struct [TYPES::EQUATIONS\\_TO\\_SOLVER\\_MATRIX\\_MAPS\\_JM\\_TYPE](#)
- struct [TYPES::EQUATIONS\\_ROW\\_TO\\_SOLVER\\_ROWS\\_MAP\\_TYPE](#)

*Contains information on the mapping from the equations rows in an equations set to the solver rows.*

- struct [TYPES::EQUATIONS\\_SET\\_TO\\_SOLVER\\_MAP\\_TYPE](#)

*Contains information on the mappings from an equations set to the solver matrices.*

- struct [TYPES::SOLVER\\_COL\\_TO\\_EQUATIONS\\_MAP\\_TYPE](#)

*Contains information about the mapping from a solver matrix column to equations matrices and variables.*

- struct [TYPES::SOLVER\\_DOF\\_TO\\_VARIABLE\\_MAP\\_TYPE](#)

*Contains information about mapping the solver dof to the field variable dofs in the equations set.*

- struct [TYPES::SOLVER\\_COL\\_TO\\_EQUATIONS\\_SET\\_MAP\\_TYPE](#)

*Contains information about the mappings from a solver matrix to the equations in an equations set.*

- struct [TYPES::SOLVER\\_COL\\_TO\\_EQUATIONS\\_SETS\\_MAP\\_TYPE](#)

*Contains information on the mappings from a solver matrix to equations sets.*

- struct [TYPES::SOLVER\\_ROW\\_TO\\_EQUATIONS\\_SET\\_MAP\\_TYPE](#)

*Contains information on the mappings from a solver row to the equations rows.*

- struct [TYPES::SOLUTION\\_MAPPING\\_CREATE\\_VALUES\\_CACHE\\_TYPE](#)

*Contains information about the cached create values for a solution mapping.*

- struct [TYPES::SOLUTION\\_MAPPING\\_TYPE](#)

*Contains information on the solution mapping between the global equation sets and the solver matrices.*

- struct [TYPES::SOLUTION\\_TYPE](#)

*Contains information on the solution of a problem.*

- struct [TYPES::SOLUTION\\_PTR\\_TYPE](#)
- struct [TYPES::PROBLEM\\_LINEAR\\_DATA\\_TYPE](#)

*Contains information on any data required for a linear solution.*
- struct [TYPES::PROBLEM\\_NONLINEAR\\_DATA\\_TYPE](#)

*Contains information on any data required for a non-linear solution.*
- struct [TYPES::PROBLEM\\_TIME\\_DATA\\_TYPE](#)

*Contains information on any data required for a time-dependent solution.*
- struct [TYPES::PROBLEM\\_CONTROL\\_TYPE](#)
- struct [TYPES::PROBLEM\\_TYPE](#)

*Contains information for a problem.*
- struct [TYPES::PROBLEM\\_PTR\\_TYPE](#)

*A buffer type to allow for an array of pointers to a [PROBLEM\\_TYPE](#).*
- struct [TYPES::PROBLEMS\\_TYPE](#)

*Contains information on the problems defined on a region.*
- struct [TYPES::REGION\\_PTR\\_TYPE](#)

*A buffer type to allow for an array of pointers to a [REGION\\_TYPE](#).*
- struct [TYPES::REGION\\_TYPE](#)

*Contains information for a region.*

## Namespaces

- namespace [TYPES](#)

*This module contains all type definitions in order to avoid cyclic module references.*

### 8.61.1 Detailed Description

#### Id

[types.f90](#) 181 2008-10-26 18:59:43Z chrispbradley

#### Author:

Chris Bradley This module contains all type definitions in order to avoid cyclic module references.

### 8.61.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [types.f90](#).

# Index

\_DP  
    KINDS\_ComplexKinds, 97

\_SP  
    KINDS\_ComplexKinds, 98

\_SYSTEM\_TYPE\_STRING  
    COORDINATE\_ROUTINES, 313

ABSOLUTE\_TOLERANCE  
    TYPES::LINEAR\_ITERATIVE\_SOLVER\_TYPE, 1248  
    TYPES::NONLINEAR\_SOLVER\_TYPE, 1280

ACCESS\_TYPE  
    BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE, 780

additional\_doc/Installation.dox, 1345

additional\_doc/library\_commands.dox, 1346

ADDITIVE\_CONSTANT  
    TYPES::SOLVER\_DOF\_TO\_VARIABLE\_MAP\_TYPE, 1318

ADDITIVE\_CONSTANTS  
    TYPES::VARIABLE\_TO\_SOLVER\_COL\_MAP\_TYPE, 1339

ADJACENT\_DOMAINS  
    TYPES::DOMAIN\_MAPPING\_TYPE, 1106

ADJACENT\_DOMAINS\_LIST  
    TYPES::DOMAIN\_MAPPING\_TYPE, 1106

ADJACENT\_DOMAINS\_PTR  
    TYPES::DOMAIN\_MAPPING\_TYPE, 1106

ADJACENT\_ELEMENTS  
    TYPES::DECOMPOSITION\_ELEMENT\_TYPE, 1055  
    TYPES::MESH\_ELEMENT\_TYPE, 1257

ADJACENT\_LINES  
    TYPES::DECOMPOSITION\_LINE\_TYPE, 1059

adjustl\_  
    ISO\_VARYING\_STRING, 499  
    ISO\_VARYING\_STRING::assignment(=), 907

adjustr\_  
    ISO\_VARYING\_STRING, 499  
    ISO\_VARYING\_STRING::adjustr, 906

ALL\_DIAG\_TYPE  
    BASE\_ROUTINES\_DiagnosticTypes, 26

ALL\_TIMING\_TYPE  
    BASE\_ROUTINES\_TimingTypes, 28

ANALYTIC  
    TYPES::EQUATIONS\_SET\_TYPE, 1180

ANALYTIC\_ANALYSIS\_CALCULATE  
    ANALYTIC\_ANALYSIS\_ROUTINES, 159

ANALYTIC\_ANALYSIS\_EXPORT  
    ANALYTIC\_ANALYSIS\_ROUTINES, 159

ANALYTIC\_ANALYSIS\_FORTRAN\_FILE\_OPEN  
    ANALYTIC\_ANALYSIS\_ROUTINES, 159

ANALYTIC\_ANALYSIS\_FORTRAN\_FILE\_WRITE\_STRING  
    ANALYTIC\_ANALYSIS\_ROUTINES, 160

ANALYTIC\_ANALYSIS\_INTEGRAL\_ABSOLUTE\_ERROR\_GET  
    ANALYTIC\_ANALYSIS\_ROUTINES, 160

ANALYTIC\_ANALYSIS\_INTEGRAL\_ANALYTIC\_VALUE\_GET  
    ANALYTIC\_ANALYSIS\_ROUTINES, 160

ANALYTIC\_ANALYSIS\_INTEGRAL\_NUMERICAL\_VALUE\_GET  
    ANALYTIC\_ANALYSIS\_ROUTINES, 161

ANALYTIC\_ANALYSIS\_INTEGRAL\_PERCENT\_ERROR\_GET  
    ANALYTIC\_ANALYSIS\_ROUTINES, 161

ANALYTIC\_ANALYSIS\_INTEGRAL\_RELATIVE\_ERROR\_GET  
    ANALYTIC\_ANALYSIS\_ROUTINES, 161

ANALYTIC\_ANALYSIS\_NODE\_ABSOLUTE\_ERROR\_GET  
    ANALYTIC\_ANALYSIS\_ROUTINES, 162

ANALYTIC\_ANALYSIS\_NODE\_ANALYTIC\_VALUE\_GET  
    ANALYTIC\_ANALYSIS\_ROUTINES, 162

ANALYTIC\_ANALYSIS\_NODE\_NUMERICAL\_VALUE\_GET  
    ANALYTIC\_ANALYSIS\_ROUTINES, 163

ANALYTIC\_ANALYSIS\_NODE\_PERCENT\_ERROR\_GET  
    ANALYTIC\_ANALYSIS\_ROUTINES, 163

ANALYTIC\_ANALYSIS\_NODE\_RELATIVE\_ERROR\_GET  
    ANALYTIC\_ANALYSIS\_ROUTINES, 164

ANALYTIC\_ANALYSIS\_RMS\_ABSOLUTE\_-  
    ERROR\_GET  
        ANALYTIC\_ANALYSIS\_ROUTINES, 164  
ANALYTIC\_ANALYSIS\_RMS\_PERCENT\_-  
    ERROR\_GET  
        ANALYTIC\_ANALYSIS\_ROUTINES, 165  
ANALYTIC\_ANALYSIS\_RMS\_RELATIVE\_-  
    ERROR\_GET  
        ANALYTIC\_ANALYSIS\_ROUTINES, 165  
ANALYTIC\_ANALYSIS\_ROUTINES, 157  
    ANALYTIC\_ANALYSIS\_CALCULATE, 159  
    ANALYTIC\_ANALYSIS\_EXPORT, 159  
    ANALYTIC\_ANALYSIS\_FORTRAN\_-  
        FILE\_OPEN, 159  
    ANALYTIC\_ANALYSIS\_FORTRAN\_-  
        FILE\_WRITE\_STRING, 160  
    ANALYTIC\_ANALYSIS\_INTEGRAL\_-  
        ABSOLUTE\_ERROR\_GET, 160  
    ANALYTIC\_ANALYSIS\_INTEGRAL\_-  
        ANALYTIC\_VALUE\_GET, 160  
    ANALYTIC\_ANALYSIS\_INTEGRAL\_-  
        NUMERICAL\_VALUE\_GET, 161  
    ANALYTIC\_ANALYSIS\_INTEGRAL\_-  
        PERCENT\_ERROR\_GET, 161  
    ANALYTIC\_ANALYSIS\_INTEGRAL\_-  
        RELATIVE\_ERROR\_GET, 161  
    ANALYTIC\_ANALYSIS\_NODE\_-  
        ABSOLUTE\_ERROR\_GET, 162  
    ANALYTIC\_ANALYSIS\_NODE\_-  
        ANALYTIC\_VALUE\_GET, 162  
    ANALYTIC\_ANALYSIS\_NODE\_-  
        NUMERICAL\_VALUE\_GET, 163  
    ANALYTIC\_ANALYSIS\_NODE\_-  
        PERCENT\_ERROR\_GET, 163  
    ANALYTIC\_ANALYSIS\_NODE\_-  
        RELATIVE\_ERROR\_GET, 164  
    ANALYTIC\_ANALYSIS\_RMS\_-  
        ABSOLUTE\_ERROR\_GET, 164  
    ANALYTIC\_ANALYSIS\_RMS\_PERCENT\_-  
        ERROR\_GET, 165  
    ANALYTIC\_ANALYSIS\_RMS\_-  
        RELATIVE\_ERROR\_GET, 165  
ANALYTIC\_FINISHED  
    TYPES::EQUATIONS\_SET\_ANALYTIC\_-  
        TYPE, 1169  
AREAS  
    TYPES::FIELD\_GEOMETRIC\_-  
        PARAMETERS\_TYPE, 1203  
BASE\_ROUTINES, 166  
    BASE\_ROUTINES\_FINALISE, 170  
    BASE\_ROUTINES\_INITIALISE, 171  
    DIAG\_ALL\_SUBROUTINES, 208  
    DIAG\_FILE\_OPEN, 208  
DIAG\_FROM\_SUBROUTINE, 208  
DIAG\_OR\_TIMING, 208  
DIAG\_ROUTINE\_LIST, 208  
DIAGNOSTICS, 208  
DIAGNOSTICS1, 208  
DIAGNOSTICS2, 209  
DIAGNOSTICS3, 209  
DIAGNOSTICS4, 209  
DIAGNOSTICS5, 209  
DIAGNOSTICS\_LEVEL1, 209  
DIAGNOSTICS\_LEVEL2, 210  
DIAGNOSTICS\_LEVEL3, 210  
DIAGNOSTICS\_LEVEL4, 210  
DIAGNOSTICS\_LEVEL5, 210  
DIAGNOSTICS\_SET\_OFF, 171  
DIAGNOSTICS\_SET\_ON, 171  
ECHO\_OUTPUT, 210  
ENTERS, 172  
ERRORS, 182  
EXITS, 194  
FLAG\_ERROR\_C, 204  
FLAG\_ERROR\_VS, 204  
FLAG\_WARNING\_C, 205  
FLAG\_WARNING\_VS, 205  
MAX\_OUTPUT\_LINES, 210  
OP\_STRING, 210  
OUTPUT\_SET\_OFF, 205  
OUTPUT\_SET\_ON, 205  
ROUTINE\_STACK, 211  
TIMING, 211  
TIMING\_ALL\_SUBROUTINES, 211  
TIMING\_FILE\_OPEN, 211  
TIMING\_FROM\_SUBROUTINE, 212  
TIMING\_ROUTINE\_LIST, 212  
TIMING\_SET\_OFF, 206  
TIMING\_SET\_ON, 206  
TIMING\_SUMMARY, 212  
TIMING\_SUMMARY\_OUTPUT, 206  
WRITE\_STR, 207  
BASE\_ROUTINES::DiagnosticTypes, 26  
BASE\_ROUTINES::FileUnits, 22  
BASE\_ROUTINES::FLAG\_ERROR, 769  
    FLAG\_ERROR\_C, 769  
    FLAG\_ERROR\_VS, 769  
BASE\_ROUTINES::FLAG\_WARNING, 771  
    FLAG\_WARNING\_C, 771  
    FLAG\_WARNING\_VS, 771  
BASE\_ROUTINES::interface, 772  
    CPUTIMER, 772  
BASE\_ROUTINES::OutputType, 19  
BASE\_ROUTINES::ROUTINE\_LIST\_ITEM\_-  
    TYPE, 773  
    NAME, 773  
    NEXT\_ROUTINE, 773

NUMBER\_OF\_INVOCATIONS, 774  
 TOTAL\_EXCLUSIVE\_CPU\_TIME, 774  
 TOTAL\_EXCLUSIVE\_SYSTEM\_TIME, 774  
 TOTAL\_INCLUSIVE\_CPU\_TIME, 774  
 TOTAL\_INCLUSIVE\_SYSTEM\_TIME, 774  
 BASE\_ROUTINES::ROUTINE\_LIST\_TYPE, 775  
     HEAD, 775  
 BASE\_ROUTINES::ROUTINE\_STACK\_ITEM\_-  
     TYPE, 776  
         DIAGNOSTICS, 776  
         EXCLUSIVE\_CPU\_TIME, 776  
         EXCLUSIVE\_SYSTEM\_TIME, 777  
         INCLUSIVE\_CPU\_TIME, 777  
         INCLUSIVE\_SYSTEM\_TIME, 777  
         NAME, 777  
         PREVIOUS\_ROUTINE, 777  
         ROUTINE\_LIST\_ITEM, 777  
         TIMING, 777  
 BASE\_ROUTINES::ROUTINE\_STACK\_TYPE,  
     779  
         STACK\_POINTER, 779  
 BASE\_ROUTINES::TimingTypes, 28  
 BASE\_ROUTINES\_DiagnosticTypes  
     ALL\_DIAG\_TYPE, 26  
     FROM\_DIAG\_TYPE, 26  
     IN\_DIAG\_TYPE, 26  
 BASE\_ROUTINES\_FileUnits  
     DIAGNOSTICS\_FILE\_UNIT, 23  
     ECHO\_FILE\_UNIT, 23  
     IO1\_FILE\_UNIT, 23  
     IO2\_FILE\_UNIT, 23  
     IO3\_FILE\_UNIT, 23  
     IO4\_FILE\_UNIT, 24  
     IO5\_FILE\_UNIT, 24  
     LEARN\_FILE\_UNIT, 24  
     OPEN\_COMFILE\_UNIT, 24  
     START\_READ\_COMFILE\_UNIT, 24  
     STOP\_READ\_COMFILE\_UNIT, 25  
     TEMPORARY\_FILE\_UNIT, 25  
     TIMING\_FILE\_UNIT, 25  
 BASE\_ROUTINES\_FINALISE  
     BASE\_ROUTINES, 170  
 BASE\_ROUTINES\_INITIALISE  
     BASE\_ROUTINES, 171  
 BASE\_ROUTINES\_OutputType  
     DIAGNOSTIC\_OUTPUT\_TYPE, 19  
     ERROR\_OUTPUT\_TYPE, 20  
     GENERAL\_OUTPUT\_TYPE, 20  
     HELP\_OUTPUT\_TYPE, 20  
     TIMING\_OUTPUT\_TYPE, 21  
 BASE\_ROUTINES\_TimingTypes  
     ALL\_TIMING\_TYPE, 28  
     FROM\_TIMING\_TYPE, 28  
     IN\_TIMING\_TYPE, 28  
 BASE\_TAG\_NUMBER  
     TYPES::DISTRIBUTED\_MATRIX\_-  
         CMISS\_TYPE, 1068  
     TYPES::DISTRIBUTED\_VECTOR\_-  
         CMISS\_TYPE, 1077  
 BASES  
     TYPES::BASIS\_FUNCTIONS\_TYPE, 1040  
     TYPES::FIELD\_INTERPOLATION\_-  
         PARAMETERS\_TYPE, 1210  
 BASIS  
     TYPES::DOMAIN\_ELEMENT\_TYPE, 1091  
     TYPES::DOMAIN\_FACE\_TYPE, 1096  
     TYPES::DOMAIN\_LINE\_TYPE, 1102  
     TYPES::GENERATED\_MESH\_-  
         REGULAR\_TYPE, 1238  
     TYPES::MESH\_ELEMENT\_TYPE, 1258  
     TYPES::QUADRATURE\_TYPE, 1297  
 BASIS\_FINISHED  
     TYPES::BASIS\_TYPE, 1045  
 BINARY\_FILE, 213  
     BINARY\_FILE\_READABLE, 225  
     BINARY\_FILE\_USED, 225  
     BINARY\_FILE\_WRITABLE, 225  
     CLOSE\_BINARY\_FILE, 214  
     CLOSE\_CMISS\_BINARY\_FILE, 214  
     CMISS\_BINARY\_FILE\_HEADER, 225  
     CMISS\_BINARY\_HISTORY\_FILE, 225  
     CMISS\_BINARY\_IDENTITY, 225  
     CMISS\_BINARY\_IDENTITY\_HEADER,  
         225  
     CMISS\_BINARY\_MACHINE\_HEADER,  
         225  
     CMISS\_BINARY\_MATRIX\_FILE, 225  
     CMISS\_BINARY\_SIGNAL\_FILE, 226  
     FILE\_BEGINNING, 226  
     FILE\_CHANGE\_ENDIAN, 226  
     FILE\_CURRENT, 226  
     FILE\_END, 226  
     FILE\_SAME\_ENDIAN, 226  
     INQUIRE\_EOF\_BINARY\_FILE, 215  
     INQUIRE\_OPEN\_BINARY\_FILE, 215  
     MAX\_NUM\_BINARY\_FILES, 226  
     OPEN\_BINARY\_FILE, 215  
     OPEN\_CMISS\_BINARY\_FILE, 215  
     READ\_BINARY\_FILE\_CHARACTER, 216  
     READ\_BINARY\_FILE\_DP, 216  
     READ\_BINARY\_FILE\_DP1, 216  
     READ\_BINARY\_FILE\_DPC, 216  
     READ\_BINARY\_FILE\_DPC1, 217  
     READ\_BINARY\_FILE\_INTG, 217  
     READ\_BINARY\_FILE\_INTG1, 217  
     READ\_BINARY\_FILE\_LINTG, 217  
     READ\_BINARY\_FILE\_LINTG1, 217  
     READ\_BINARY\_FILE\_LOGICAL, 218

READ\_BINARY\_FILE\_LOGICAL1, 218  
READ\_BINARY\_FILE\_SINTG, 218  
READ\_BINARY\_FILE\_SINTG1, 218  
READ\_BINARY\_FILE\_SP, 219  
READ\_BINARY\_FILE\_SP1, 219  
READ\_BINARY\_FILE\_SPC, 219  
READ\_BINARY\_FILE\_SPC1, 219  
READ\_BINARY\_TAG\_HEADER, 219  
RESET\_BINARY\_NUMBER\_TAGS, 220  
SET\_BINARY\_FILE, 220  
SKIP\_BINARY\_FILE, 220  
SKIP\_BINARY\_TAGS, 220  
SKIP\_CM\_BINARY\_HEADER, 220  
WRITE\_BINARY\_FILE\_CHARACTER, 221  
WRITE\_BINARY\_FILE\_DP, 221  
WRITE\_BINARY\_FILE\_DP1, 221  
WRITE\_BINARY\_FILE\_DPC, 221  
WRITE\_BINARY\_FILE\_DPC1, 222  
WRITE\_BINARY\_FILE\_INTG, 222  
WRITE\_BINARY\_FILE\_INTG1, 222  
WRITE\_BINARY\_FILE\_LINTG, 222  
WRITE\_BINARY\_FILE\_LINTG1, 222  
WRITE\_BINARY\_FILE\_LOGICAL, 223  
WRITE\_BINARY\_FILE\_LOGICAL1, 223  
WRITE\_BINARY\_FILE\_SINTG, 223  
WRITE\_BINARY\_FILE\_SINTG1, 223  
WRITE\_BINARY\_FILE\_SP, 224  
WRITE\_BINARY\_FILE\_SP1, 224  
WRITE\_BINARY\_FILE\_SPC, 224  
WRITE\_BINARY\_FILE\_SPC1, 224  
WRITE\_BINARY\_TAG\_HEADER, 224  
BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE,  
    780  
    ACCESS\_TYPE, 780  
    BINARY\_FILE\_REVISION, 780  
    CHAR\_FORMAT, 780  
    CHARACTER\_SIZE, 780  
    DP\_FORMAT, 780  
    DP\_REAL\_SIZE, 781  
    DPC\_REAL\_SIZE, 781  
    ENDIAN\_TYPE, 781  
    FILE\_NAME, 781  
    FILE\_NUMBER, 781  
    INT\_FORMAT, 781  
    INTEGER\_SIZE, 781  
    LINTEGER\_SIZE, 781  
    LOGICAL\_SIZE, 781  
    MACHINE\_TYPE, 781  
    OS\_TYPE, 781  
    SINTEGER\_SIZE, 782  
    SP\_FORMAT, 782  
    SP\_REAL\_SIZE, 782  
    SPC\_REAL\_SIZE, 782  
BINARY\_FILE::BINARY\_FILE\_TYPE, 783  
FILE\_INFORMATION, 783  
BINARY\_FILE::BINARY\_TAG\_TYPE, 784  
HEADER, 784  
INDEX, 784  
NUM\_BYTES, 784  
NUM\_HEADER\_BYTES, 784  
NUM\_SUBTAGS, 784  
BINARY\_FILE::interface, 785  
    BINARYCLOSEFILE, 785  
    BINARYOPENFILE, 785  
    BINARYSETFILE, 785  
    BINARYSKIPFILE, 785  
    ISBINARYFILEOPEN, 785  
    ISENDBINARYFILE, 786  
BINARY\_FILE::READ\_BINARY\_FILE, 787  
    READ\_BINARY\_FILE\_CHARACTER, 787  
    READ\_BINARY\_FILE\_DP, 787  
    READ\_BINARY\_FILE\_DP1, 787  
    READ\_BINARY\_FILE\_DPC, 787  
    READ\_BINARY\_FILE\_DPC1, 788  
    READ\_BINARY\_FILE\_INTG, 788  
    READ\_BINARY\_FILE\_INTG1, 788  
    READ\_BINARY\_FILE\_LINTG, 788  
    READ\_BINARY\_FILE\_LINTG1, 788  
    READ\_BINARY\_FILE\_LOGICAL, 788  
    READ\_BINARY\_FILE\_LOGICAL1, 788  
    READ\_BINARY\_FILE\_SINTG, 789  
    READ\_BINARY\_FILE\_SINTG1, 789  
    READ\_BINARY\_FILE\_SP, 789  
    READ\_BINARY\_FILE\_SP1, 789  
    READ\_BINARY\_FILE\_SPC, 789  
    READ\_BINARY\_FILE\_SPC1, 789  
BINARY\_FILE::WRITE\_BINARY\_FILE, 791  
    WRITE\_BINARY\_FILE\_CHARACTER, 791  
    WRITE\_BINARY\_FILE\_DP, 791  
    WRITE\_BINARY\_FILE\_DP1, 791  
    WRITE\_BINARY\_FILE\_DPC, 791  
    WRITE\_BINARY\_FILE\_DPC1, 792  
    WRITE\_BINARY\_FILE\_INTG, 792  
    WRITE\_BINARY\_FILE\_INTG1, 792  
    WRITE\_BINARY\_FILE\_LINTG, 792  
    WRITE\_BINARY\_FILE\_LINTG1, 792  
    WRITE\_BINARY\_FILE\_LOGICAL, 792  
    WRITE\_BINARY\_FILE\_LOGICAL1, 792  
    WRITE\_BINARY\_FILE\_SINTG, 793  
    WRITE\_BINARY\_FILE\_SINTG1, 793  
    WRITE\_BINARY\_FILE\_SP, 793  
    WRITE\_BINARY\_FILE\_SP1, 793  
    WRITE\_BINARY\_FILE\_SPC, 793  
    WRITE\_BINARY\_FILE\_SPC1, 793  
binary\_file\_c.c  
    BinaryCloseFile, 1359  
    binaryfiles, 1360  
    BinaryOpenFile, 1359

BinaryReadFile, 1359  
 BinarySetFile, 1359  
 BinarySkipFile, 1359  
 BinaryWriteFile, 1360  
 CHARTYPE, 1357  
 COMPLEXTYPE, 1357  
 DOUBLECOMPLEXTYPE, 1358  
 DOBULETYPE, 1358  
 FLIPENDIAN, 1358  
 FLOATTYPE, 1358  
 INTEGERTYPE, 1358  
 IsBinaryFileOpen, 1360  
 IsEndBinaryFile, 1360  
 logical, 1359  
 LOGICALTYPE, 1358  
 LONGINTTYPE, 1358  
 MAXBINFILES, 1358  
 QUADRUPLECOMPLEXTYPE, 1358  
 QUADRUPLETYPY, 1358  
 SAMEENDIAN, 1359  
 SHORTINTTYPE, 1359  
**BINARY\_FILE\_READABLE**  
 BINARY\_FILE, 225  
**BINARY\_FILE\_REVISION**  
 BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE, 780  
**BINARY\_FILE\_USED**  
 BINARY\_FILE, 225  
**BINARY\_FILE\_WRITABLE**  
 BINARY\_FILE, 225  
**BINARYCLOSEFILE**  
 BINARY\_FILE::interface, 785  
**BinaryCloseFile**  
 binary\_file\_c.c, 1359  
**binaryfiles**  
 binary\_file\_c.c, 1360  
**BINARYOPENFILE**  
 BINARY\_FILE::interface, 785  
**BinaryOpenFile**  
 binary\_file\_c.c, 1359  
**BinaryReadFile**  
 binary\_file\_c.c, 1359  
**BINARYSETFILE**  
 BINARY\_FILE::interface, 785  
**BinarySetFile**  
 binary\_file\_c.c, 1359  
**BINARYSKIPFILE**  
 BINARY\_FILE::interface, 785  
**BinarySkipFile**  
 binary\_file\_c.c, 1359  
**BinaryWriteFile**  
 binary\_file\_c.c, 1360  
**BLAS**, 227  
 BLAS::interface, 795  
 DASUM, 795  
 DAXPY, 795  
 DCOPY, 795  
 DDOT, 795  
 DNRM2, 796  
 DROT, 796  
 DROTG, 796  
 DSCAL, 796  
 DTRSV, 796  
 SASUM, 796  
 SAXPY, 796  
 SCOPY, 796  
 SDOT, 797  
 SNRM2, 797  
 SROT, 797  
 SROTG, 797  
 SSCAL, 797  
 STRSV, 797  
**BLOCK\_LENGTHS**  
 COMP\_ENVIRONMENT::MPI\_COMPUTATIONAL\_NODE\_TYPE, 826  
**BOUNDARY\_CONDITIONS**  
 TYPES::EQUATIONS\_SET\_FIXED\_CONDITIONS\_TYPE, 1171  
**BOUNDARY\_LIST**  
 TYPES::DOMAIN\_MAPPING\_TYPE, 1107  
**BUBBLE\_SORT\_DP**  
 SORTING, 723  
 SORTING::BUBBLE\_SORT, 1014  
**BUBBLE\_SORT\_INTG**  
 SORTING, 723  
 SORTING::BUBBLE\_SORT, 1014  
**BUBBLE\_SORT\_SP**  
 SORTING, 723  
 SORTING::BUBBLE\_SORT, 1014  
**C2FSTRING**  
 F90C, 350  
**char\_auto**  
 ISO\_VARYING\_STRING, 500  
 ISO\_VARYING\_STRING::char, 911  
**char\_fixed**  
 ISO\_VARYING\_STRING, 500  
 ISO\_VARYING\_STRING::char, 911  
**CHAR\_FORMAT**  
 BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE, 780  
**CHARACTER\_SIZE**  
 BINARY\_FILE::BINARY\_FILE\_INFO\_TYPE, 780  
 MACHINE\_CONSTANTS, 540  
**CHARACTER\_TO\_LOWERCASE\_C**  
 STRINGS, 730

STRINGS::CHARACTER\_TO\_-
   
 LOWERCASE, 1017

CHARACTER\_TO\_LOWERCASE\_VS
   
 STRINGS, 730

STRINGS::CHARACTER\_TO\_-
   
 LOWERCASE, 1017

CHARACTER\_TO\_UPPERCASE\_C
   
 STRINGS, 730

STRINGS::CHARACTER\_TO\_-
   
 UPPERCASE, 1018

CHARACTER\_TO\_UPPERCASE\_VS
   
 STRINGS, 731

STRINGS::CHARACTER\_TO\_-
   
 UPPERCASE, 1018

chars

ISO\_VARYING\_STRING::VARYING\_-
   
 STRING, 936

CHARTYPE

binary\_file\_c.c, 1357

CL

CLASSICAL\_FIELD\_ROUTINES, 229

CLASS

TYPES::EQUATIONS\_SET\_TYPE, 1180

TYPES::PROBLEM\_TYPE, 1290

CLASSICAL\_FIELD\_EQUATIONS\_SET\_SETUP

CLASSICAL\_FIELD\_ROUTINES, 229

CLASSICAL\_FIELDFINITE\_ELEMENT\_-

CALCULATE
   
 CLASSICAL\_FIELD\_ROUTINES, 230

CLASSICAL\_FIELDFINITE\_ELEMENT\_-

JACOBIAN\_EVALUATE
   
 CLASSICAL\_FIELD\_ROUTINES, 231

CLASSICAL\_FIELDFINITE\_ELEMENT\_-

RESIDUAL\_EVALUATE
   
 CLASSICAL\_FIELD\_ROUTINES, 231

CLASSICAL\_FIELD\_PROBLEM\_CLASS\_-

TYPE\_GET
   
 CLASSICAL\_FIELD\_ROUTINES, 232

CLASSICAL\_FIELD\_PROBLEM\_CLASS\_-

TYPE\_SET
   
 CLASSICAL\_FIELD\_ROUTINES, 232

CLASSICAL\_FIELD\_PROBLEM\_SETUP

CLASSICAL\_FIELD\_ROUTINES, 233

CLASSICAL\_FIELD\_ROUTINES, 228

CL, 229

CLASSICAL\_FIELD\_EQUATIONS\_SET\_-
   
 SETUP, 229

CLASSICAL\_FIELDFINITE\_ELEMENT\_-

CALCULATE, 230

CLASSICAL\_FIELDFINITE\_ELEMENT\_-

JACOBIAN\_EVALUATE, 231

CLASSICAL\_FIELDFINITE\_ELEMENT\_-

RESIDUAL\_EVALUATE, 231

CLASSICAL\_FIELD\_ROUTINES, 232

MPI\_ERROR\_CHECK, 236

PARMETIS, 237

PARMETIS\_PARTKWAY, 237

PARMETIS\_PARTMESHKWAY, 237

ParMETIS\_V3\_PartK, 798

ParMETIS\_V3\_PartMeshKway, 798

CLASSICAL\_FIELD\_PROBLEM\_CLASS\_-
   
 TYPE\_GET, 232

CLASSICAL\_FIELD\_PROBLEM\_CLASS\_-
   
 TYPE\_SET, 232

CLASSICAL\_FIELD\_PROBLEM\_SETUP,
   
 233

CLOSE\_BINARY\_FILE

BINARY\_FILE, 214

CLOSE\_CMISS\_BINARY\_FILE

BINARY\_FILE, 214

CMISS, 234

CMISS\_BUILD\_VERSION, 235

CMISS\_FINALISE, 234

CMISS\_INITIALISE, 234

CMISS\_MAJOR\_VERSION, 235

CMISS\_MINOR\_VERSION, 235

CMISS\_REVISION\_VERSION, 235

CMISS\_WRITE\_ERROR, 235

TYPES::DISTRIBUTED\_MATRIX\_TYPE,
   
 1073

TYPES::DISTRIBUTED\_VECTOR\_TYPE,
   
 1085

CMISS\_BINARY\_FILE\_HEADER

BINARY\_FILE, 225

CMISS\_BINARY\_HISTORY\_FILE

BINARY\_FILE, 225

CMISS\_BINARY\_IDENTITY

BINARY\_FILE, 225

CMISS\_BINARY\_IDENTITY\_HEADER

BINARY\_FILE, 225

CMISS\_BINARY\_MACHINE\_HEADER

BINARY\_FILE, 225

CMISS\_BINARY\_MATRIX\_FILE

BINARY\_FILE, 225

CMISS\_BINARY\_SIGNAL\_FILE

BINARY\_FILE, 226

CMISS\_BUILD\_VERSION

CMISS, 235

CMISS\_FINALISE

CMISS, 234

CMISS\_INITIALISE

CMISS, 234

CMISS\_MAJOR\_VERSION

CMISS, 235

CMISS\_MINOR\_VERSION

CMISS, 235

CMISS\_MPI, 236

MPI\_ERROR\_CHECK, 236

PARMETIS, 237

PARMETIS\_PARTKWAY, 237

PARMETIS\_PARTMESHKWAY, 237

ParMETIS\_V3\_PartK, 798

ParMETIS\_V3\_PartMeshKway, 798

CMISS\_PETSC, 239  
 PETSC\_ERRORHANDLING\_SET\_OFF, 246  
 PETSC\_ERRORHANDLING\_SET\_ON, 246  
 PETSC\_FINALIZE, 246  
 PETSC\_HANDLE\_ERROR, 291  
 PETSC\_INITIALIZE, 247  
 PETSC\_ISCOLORINGDESTROY, 247  
 PETSC\_ISCOLORINGFINALISE, 248  
 PETSC\_ISCOLORINGINITIALISE, 248  
 PETSC\_ISDESTROY, 248  
 PETSC\_ISFINALISE, 249  
 PETSC\_ISINITIALISE, 249  
 PETSC\_ISLOCALTOGLOBALMAPPINGAPPLY, 249  
 PETSC\_ISLOCALTOGLOBALMAPPINGAPPLYIS, 250  
 PETSC\_ISLOCALTOGLOBALMAPPINGCREATE, 250  
 PETSC\_ISLOCALTOGLOBALMAPPINGDESTROY, 250  
 PETSC\_ISLOCALTOGLOBALMAPPINGFINALISE, 251  
 PETSC\_ISLOCALTOGLOBALMAPPINGINITIALISE, 251  
 PETSC\_KSPCREATE, 251  
 PETSC\_KSPDESTROY, 252  
 PETSC\_KSPFINALISE, 252  
 PETSC\_KSPGETCONVERGEDREASON, 253  
 PETSC\_KSPGETITERATIONNUMBER, 253  
 PETSC\_KSPGETPC, 253  
 PETSC\_KSPGETRESIDUALNORM, 254  
 PETSC\_KSPINITIALISE, 254  
 PETSC\_KSPSETFROMOPTIONS, 255  
 PETSC\_KSPSETOPERATORS, 255  
 PETSC\_KSPSETTOLERANCES, 256  
 PETSC\_KSPSETTYPE, 256  
 PETSC\_KSPSETUP, 256  
 PETSC\_KSPSOLVE, 257  
 PETSC\_LOGPRINTSUMMARY, 257  
 PETSC\_MATASSEMBLYBEGIN, 258  
 PETSC\_MATASSEMBLYEND, 258  
 PETSC\_MATCREATE, 258  
 PETSC\_MATCREATEMPIAIJ, 259  
 PETSC\_MATCREATEMPIDENSE, 259  
 PETSC\_MATCREATESEQAIJ, 260  
 PETSC\_MATCREATESEQDENSE, 260  
 PETSC\_MATDESTROY, 261  
 PETSC\_MATFDCOLORINGCREATE, 261  
 PETSC\_MATFDCOLORINGDESTROY, 262  
 PETSC\_MATFDCOLORINGFINALISE, 262  
 PETSC\_MATFDCOLORINGINITIALISE, 262  
 PETSC\_MATFDCOLORINGSETFROMOPTIONS, 263  
 PETSC\_MATFINALISE, 263  
 PETSC\_MATGETARRAY, 263  
 PETSC\_MATGETCOLORING, 264  
 PETSC\_MATGETOWNERSHIPRANGE, 264  
 PETSC\_MATGETVALUES, 265  
 PETSC\_MATINITIALISE, 265  
 PETSC\_MATRESTOREARRAY, 265  
 PETSC\_MATSETLOCALTOGLOBALMAPPING, 266  
 PETSC\_MATSETOPTION, 266  
 PETSC\_MATSETSIZES, 267  
 PETSC\_MATSETVALUE, 267  
 PETSC\_MATSETVALUELOCAL, 267  
 PETSC\_MATSETVALUES, 268  
 PETSC\_MATSETVALUESLOCAL, 268  
 PETSC\_MATVIEW, 269  
 PETSC\_MATZEROENTRIES, 269  
 PETSC\_PCFINALISE, 270  
 PETSC\_PCINITIALISE, 270  
 PETSC\_PCSETTYPE, 270  
 PETSC\_SNES\_LINESEARCH\_CUBIC, 291  
 PETSC\_SNES\_LINESEARCH\_NO, 291  
 PETSC\_SNES\_LINESEARCH\_NONORMS, 291  
 PETSC\_SNES\_LINESEARCH\_QUADRATIC, 291  
 PETSC\_SNESCREATE, 271  
 PETSC\_SNESDESTROY, 271  
 PETSC\_SNESFINALISE, 272  
 PETSC\_SNESGETCONVERGEDREASON, 272  
 PETSC\_SNESGETFUNCTIONNORM, 272  
 PETSC\_SNESGETITERATIONNUMBER, 273  
 PETSC\_SNESINITIALISE, 273  
 PETSC\_SNESLINESEARCHSET, 274  
 PETSC\_SNESLINESEARCHSETPARAMS, 274  
 PETSC\_SNESSETFROMOPTIONS, 275  
 PETSC\_SNESSETFUNCTION, 275  
 PETSC\_SNESSETJACOBIAN\_MATFDCOLORING, 276  
 PETSC\_SNESSETJACOBIAN\_SOLVER, 276  
 PETSC\_SNESSETTOLERANCES, 276  
 PETSC\_SNESSETTRUSTREGIONTOLERANCE, 277  
 PETSC\_SNESSETTYPE, 277  
 PETSC\_SNESSOLVE, 278  
 PETSC\_VECASSEMBLYBEGIN, 278  
 PETSC\_VECASSEMBLYEND, 279  
 PETSC\_VECCREATE, 279

PETSC\_VECCREATEGHOST, 279  
PETSC\_VECCREATEGHOSTWITHARRAY,  
    280  
PETSC\_VECCREATEMPI, 280  
PETSC\_VECCREATEMPIWITHARRAY,  
    281  
PETSC\_VECCREATESEQ, 281  
PETSC\_VECCREATESEQWITHARRAY,  
    282  
PETSC\_VECDESTROY, 282  
PETSC\_VECDUPLICATE, 282  
PETSC\_VECFINALISE, 283  
PETSC\_VECGETARRAY, 283  
PETSC\_VECGETARRAYF90, 283  
PETSC\_VECGETLOCALSIZE, 284  
PETSC\_VECGETOWNERSHIPRANGE, 284  
PETSC\_VECGETSIZE, 285  
PETSC\_VECGETVALUES, 285  
PETSC\_VECGHOSTGETLOCALFORM,  
    285  
PETSC\_VECGHOSTRESTORELOCALFORM,  
    286  
PETSC\_VECGHOSTUPDATEBEGIN, 286  
PETSC\_VECGHOSTUPDATEEND, 287  
PETSC\_VECINITIALISE, 287  
PETSC\_VCRESTOREARRAY, 287  
PETSC\_VCRESTOREARRAYF90, 288  
PETSC\_VECSET, 288  
PETSC\_VECSETFROMOPTIONS, 288  
PETSC\_VECSETLOCALTOGLOBALMAPPING,  
    289  
PETSC\_VECSETSIZES, 289  
PETSC\_VECSETVALUES, 290  
PETSC\_VECSETVALUESLOCAL, 290  
PETSC\_VECVIEW, 291  
CMISS\_PETSC::interface, 799  
    ISColoringDestroy, 801  
    ISDestroy, 801  
    ISLocalToGlobalMappingApply, 801  
    ISLocalToGlobalMappingApplyIS, 801  
    ISLocalToGlobalMappingCreate, 801  
    ISLocalToGlobalMappingDestroy, 801  
    KSPCreate, 801  
    KSPDestroy, 801  
    KSPGetConvergedReason, 801  
    KSPGetIterationNumber, 801  
    KSPGetPC, 802  
    KSPGetResidualNorm, 802  
    KSPSetFromOptions, 802  
    KSPSetOperators, 802  
    KSPSetTolerances, 802  
    KSPSetType, 802  
    KSPSetUp, 802  
    KSPSolve, 802  
    MatAssemblyBegin, 802  
    MatAssemblyEnd, 802  
    MatCreate, 803  
    MatCreateMPIAIJ, 803  
    MatCreateMPIDense, 803  
    MatCreateSeqAIJ, 803  
    MatCreateSeqDense, 803  
    MatDestroy, 803  
    MatFDColoringCreate, 803  
    MatFDColoringDestroy, 803  
    MatFDColoringSetFromOptions, 803  
    MatGetArray, 803  
    MatGetArrayF90, 804  
    MatGetColoring, 804  
    MatGetOwnershipRange, 804  
    MatGetValues, 804  
    MatRestoreArray, 804  
    MatRestoreArrayF90, 804  
    MatSetLocalToGlobalMapping, 804  
    MatSetOption, 804  
    MatSetSizes, 804  
    MatSetValue, 804  
    MatSetValueLocal, 805  
    MatSetValues, 805  
    MatSetValuesLocal, 805  
    MatView, 805  
    MatZeroEntries, 805  
    PCsetType, 805  
    PetscFinalize, 805  
    PetscInitialize, 805  
    PetscLogPrintSummary, 805  
    SNESCreate, 805  
    SNESDestroy, 806  
    SNESGetConvergedReason, 806  
    SNESGetFunctionNorm, 806  
    SNESGetIterationNumber, 806  
    SNESLineSearchSet, 806  
    SNESLineSearchSetParams, 806  
    SNESSetFromOptions, 806  
    SNESSetFunction, 806  
    SNESSetTolerances, 806  
    SNESSetTrustRegionTolerance, 806  
    SNESSetType, 807  
    SNESolve, 807  
    VecAssemblyBegin, 807  
    VecAssemblyEnd, 807  
    VecCreate, 807  
    VecCreateGhost, 807  
    VecCreateGhostWithArray, 807  
    VecCreateMPI, 807  
    VecCreateMPIWithArray, 807  
    VecCreateSeq, 807  
    VecCreateSeqWithArray, 808  
    VecDestroy, 808

VecDuplicate, 808  
 VecGetArray, 808  
 VecGetArrayF90, 808  
 VecGetLocalSize, 808  
 VecGetOwnershipRange, 808  
 VecGetSize, 808  
 VecGetValues, 808  
 VecGhostGetLocalForm, 808  
 VecGhostRestoreLocalForm, 809  
 VecGhostUpdateBegin, 809  
 VecGhostUpdateEnd, 809  
 VecRestoreArray, 809  
 VecRestoreArrayF90, 809  
 VecSet, 809  
 VecSetFromOptions, 809  
 VecSetLocalToGlobalMapping, 809  
 VecSetSizes, 809  
 VecSetValue, 809  
 VecSetValuesLocal, 810  
 VecView, 810  
**CMISS\_PETSC::PETSC\_SNESSETJACOBIAN,**  
     811  
**PETSC\_SNESSETJACOBIAN\_-**  
     MATFDCOLORING, 811  
**PETSC\_SNESSETJACOBIAN\_SOLVER,**  
     811  
**CMISS\_PETSC\_TYPES**, 293  
**CMISS\_PETSC\_TYPES::PETSC\_IS\_TYPE**, 812  
**CMISS\_PETSC\_TYPES::PETSC\_-**  
     ISCOLORING\_TYPE, 813  
**CMISS\_PETSC\_TYPES::PETSC\_-**  
     ISLOCALTOGLOBALMAPPING\_-  
     TYPE, 814  
**CMISS\_PETSC\_TYPES::PETSC\_KSP\_TYPE**,  
     815  
**CMISS\_PETSC\_TYPES::PETSC\_MAT\_TYPE**,  
     816  
**CMISS\_PETSC\_TYPES::PETSC\_-**  
     MATFDCOLORING\_TYPE, 817  
**CMISS\_PETSC\_TYPES::PETSC\_PC\_TYPE**, 818  
**CMISS\_PETSC\_TYPES::PETSC\_SNES\_TYPE**,  
     819  
**CMISS\_PETSC\_TYPES::PETSC\_VEC\_TYPE**,  
     820  
**CMISS\_REVISION\_VERSION**  
     CMISS, 235  
**CMISS\_VECTOR**  
     TYPES::DISTRIBUTED\_VECTOR\_-  
         TRANSFER\_TYPE, 1082  
**CMISS\_WRITE\_ERROR**  
     CMISS, 235  
**CO**  
     COORDINATE\_ROUTINES, 302, 303  
**COLLAPSED\_XI**

TYPES::BASIS\_TYPE, 1045  
**COLOUR**  
     TREES::TREE\_NODE\_TYPE, 1036  
**COLUMN\_DOF**  
     TYPES::VARIABLE\_TO\_EQUATIONS\_-  
         COLUMN\_MAP\_TYPE, 1334  
**COLUMN\_DOFS**  
     TYPES::ELEMENT\_MATRIX\_TYPE, 1122  
**COLUMN\_DOFS\_MAPPING**  
     TYPES::EQUATIONS\_JACOBIAN\_TO\_-  
         VARIABLE\_MAP\_TYPE, 1131  
     TYPES::EQUATIONS\_MATRIX\_TO\_-  
         VARIABLE\_MAP\_TYPE, 1163  
     TYPES::SOLVER\_COL\_TO\_EQUATIONS\_-  
         SETS\_MAP\_TYPE, 1316  
**COLUMN\_DOMAIN\_MAPPING**  
     TYPES::DISTRIBUTED\_MATRIX\_TYPE,  
         1073  
**COLUMN\_INDICES**  
     TYPES::DISTRIBUTED\_MATRIX\_-  
         PETSC\_TYPE, 1070  
     TYPES::MATRIX\_TYPE, 1253  
**COLUMN\_NUMBERS**  
     TYPES::VARIABLE\_TO\_SOLVER\_COL\_-  
         MAP\_TYPE, 1339  
**COLUMN\_TO\_DOF\_MAP**  
     TYPES::EQUATIONS\_MATRIX\_TO\_-  
         VARIABLE\_MAP\_TYPE, 1163  
**COMP\_ENVIRONMENT**, 294  
     COMPUTATIONAL\_ENVIRONMENT, 299  
     COMPUTATIONAL\_ENVIRONMENT\_-  
         FINALISE, 295  
     COMPUTATIONAL\_ENVIRONMENT\_-  
         INITIALISE, 295  
     COMPUTATIONAL\_NODE\_FINALISE, 296  
     COMPUTATIONAL\_NODE\_INITIALISE,  
         296  
     COMPUTATIONAL\_NODE\_MPI\_TYPE\_-  
         FINALISE, 297  
     COMPUTATIONAL\_NODE\_MPI\_TYPE\_-  
         INITIALISE, 297  
     COMPUTATIONAL\_NODE\_NUMBER\_-  
         GET, 297  
     COMPUTATIONAL\_NODES\_NUMBER\_-  
         GET, 298  
     MPI\_COMPUTATIONAL\_NODE\_TYPE\_-  
         DATA, 299  
**COMP\_ENVIRONMENT::CACHE\_TYPE**, 821  
     NUMBER\_LEVELS, 821  
     SIZE, 821  
**COMP\_ENVIRONMENT::COMPUTATIONAL\_-**  
     ENVIRONMENT\_TYPE, 822  
     COMPUTATIONAL\_NODES, 822  
     MPI\_COMM, 822

MY\_COMPUTATIONAL\_NODE\_-  
NUMBER, 822  
NUMBER\_COMPUTATIONAL\_NODES,  
823  
COMP\_ENVIRONMENT::COMPUTATIONAL\_-  
NODE\_TYPE, 824  
NODE\_NAME, 824  
NODE\_NAME\_LENGTH, 824  
NUMBER\_PROCESSORS, 824  
RANK, 824  
COMP\_ENVIRONMENT::MPI\_-  
COMPUTATIONAL\_NODE\_TYPE,  
826  
BLOCK\_LENGTHS, 826  
DISPLACEMENTS, 826  
MPI\_TYPE, 826  
NUM\_BLOCKS, 826  
TYPES, 827  
COMPLEXTYPE  
binary\_file\_c.c, 1357  
COMPONENT\_NUMBER  
TYPES::FIELD\_VARIABLE\_-  
COMPONENT\_TYPE, 1230  
COMPONENTS  
FIELD\_IO\_ROUTINES::FIELD\_IO\_INFO\_-  
SET, 847  
TYPES::FIELD\_VARIABLE\_TYPE, 1234  
COMPUTATIONAL\_ENVIRONMENT  
COMP\_ENVIRONMENT, 299  
COMPUTATIONAL\_ENVIRONMENT\_-  
FINALISE  
COMP\_ENVIRONMENT, 295  
COMPUTATIONAL\_ENVIRONMENT\_-  
INITIALISE  
COMP\_ENVIRONMENT, 295  
COMPUTATIONAL\_NODE\_FINALISE  
COMP\_ENVIRONMENT, 296  
COMPUTATIONAL\_NODE\_INITIALISE  
COMP\_ENVIRONMENT, 296  
COMPUTATIONAL\_NODE\_MPI\_TYPE\_-  
FINALISE  
COMP\_ENVIRONMENT, 297  
COMPUTATIONAL\_NODE\_MPI\_TYPE\_-  
INITIALISE  
COMP\_ENVIRONMENT, 297  
COMPUTATIONAL\_NODE\_NUMBER\_GET  
COMP\_ENVIRONMENT, 297  
COMPUTATIONAL\_NODES  
COMP\_ENVIRONMENT::COMPUTATIONAL\_-  
ENVIRONMENT\_TYPE, 822  
COMPUTATIONAL\_NODES\_NUMBER\_GET  
COMP\_ENVIRONMENT, 298  
CONSTANT\_DOF2PARAM\_MAP  
TYPES::FIELD\_DOF\_TO\_PARAM\_MAP\_-  
TYPE, 1200  
CONSTANT\_PARAM2DOF\_MAP  
TYPES::FIELD\_PARAM\_TO\_DOF\_MAP\_-  
TYPE, 1214  
CONTROL  
TYPES::PROBLEM\_TYPE, 1290  
CONTROL\_FINISHED  
TYPES::PROBLEM\_CONTROL\_TYPE,  
1285  
CONTROL\_TYPE  
TYPES::PROBLEM\_CONTROL\_TYPE,  
1285  
COORDINAT  
COORDINATE\_ROUTINES, 313  
COORDINATE\_CONVERT\_FROM\_RC\_DP  
COORDINATE\_ROUTINES, 303  
COORDINATE\_-  
ROUTINES::COORDINATE\_-  
CONVERT\_FROM\_RC, 828  
COORDINATE\_CONVERT\_FROM\_RC\_SP  
COORDINATE\_ROUTINES, 303  
COORDINATE\_-  
ROUTINES::COORDINATE\_-  
CONVERT\_FROM\_RC, 828  
COORDINATE\_CONVERT\_TO\_RC\_DP  
COORDINATE\_ROUTINES, 303  
COORDINATE\_-  
ROUTINES::COORDINATE\_-  
CONVERT\_TO\_RC, 829  
COORDINATE\_CONVERT\_TO\_RC\_SP  
COORDINATE\_ROUTINES, 304  
COORDINATE\_-  
ROUTINES::COORDINATE\_-  
CONVERT\_TO\_RC, 829  
COORDINATE\_CYCLINDRICAL\_POLAR\_-  
TYPE  
COORINDATE\_ROUTINES\_-  
CoordinateSystemTypes, 30  
COORDINATE\_DELTA\_CALCULATE\_DP  
COORDINATE\_ROUTINES, 304  
COORDINATE\_-  
ROUTINES::COORDINATE\_DELTA\_-  
CALCULATE, 830  
COORDINATE\_DERIVATIVE\_CONVERT\_TO\_-  
RC\_DP  
COORDINATE\_-  
ROUTINES::COORDINATE\_-  
DERIVATIVE\_CONVERT\_TO\_RC,  
831  
COORDINATE\_DERIVATIVE\_CONVERT\_TO\_-  
RC\_SP  
COORDINATE\_-  
ROUTINES::COORDINATE\_-

DERIVATIVE\_CONVERT\_TO\_RC,  
 831  
 COORDINATE\_DERIVATIVE\_NORM  
   COORDINATE\_ROUTINES, 304  
 COORDINATE\_INTERPOLATION\_ADJUST  
   COORDINATE\_ROUTINES, 305  
 COORDINATE\_INTERPOLATION\_-  
   PARAMETERS\_ADJUST  
   COORDINATE\_ROUTINES, 305  
 COORDINATE\_JACOBIAN\_AREA\_TYPE  
   COORDINATE\_ROUTINES\_JacobianType,  
   36  
 COORDINATE\_JACOBIAN\_LINE\_TYPE  
   COORDINATE\_ROUTINES\_JacobianType,  
   36  
 COORDINATE\_JACOBIAN\_VOLUME\_TYPE  
   COORDINATE\_ROUTINES\_JacobianType,  
   36  
 COORDINATE\_METRICS\_CALCULATE  
   COORDINATE\_ROUTINES, 305  
 COORDINATE\_NO\_RADIAL\_-  
   INTERPOLATION\_TYPE  
   COORDINATE\_ROUTINES\_-  
     RadialInterpolations, 34  
 COORDINATE\_OBLATE\_SPHEROIDAL\_TYPE  
   COORDINATE\_ROUTINES\_-  
     CoordinateSystemTypes, 31  
 COORDINATE\_PROLATE\_SPHEROIDAL\_-  
   TYPE  
   COORDINATE\_ROUTINES\_-  
     CoordinateSystemTypes, 31  
 COORDINATE\_RADIAL\_CUBED\_-  
   INTERPOLATION\_TYPE  
   COORDINATE\_ROUTINES\_-  
     RadialInterpolations, 34  
 COORDINATE\_RADIAL\_INTERPOLATION\_-  
   TYPE  
   COORDINATE\_ROUTINES\_-  
     RadialInterpolations, 35  
 COORDINATE\_RADIAL\_SQUARED\_-  
   INTERPOLATION\_TYPE  
   COORDINATE\_ROUTINES\_-  
     RadialInterpolations, 35  
 COORDINATE\_RECTANGULAR\_-  
   CARTESIAN\_TYPE  
   COORDINATE\_ROUTINES\_-  
     CoordinateSystemTypes, 32  
 COORDINATE\_ROUTINES, 300  
   \_SYSTEM\_TYPE\_STRING, 313  
 CO, 302, 303  
 COORDINAT, 313  
 COORDINATE\_CONVERT\_FROM\_RC\_DP,  
 303  
 COORDINATE\_CONVERT\_FROM\_RC\_SP,  
 303  
 COORDINATE\_CONVERT\_TO\_RC\_DP,  
 303  
 COORDINATE\_CONVERT\_TO\_RC\_SP, 304  
 COORDINATE\_DELTA\_CALCULATE\_DP,  
 304  
 COORDINATE\_DERIVATIVE\_NORM, 304  
 COORDINATE\_INTERPOLATION\_-  
   ADJUST, 305  
 COORDINATE\_INTERPOLATION\_-  
   PARAMETERS\_ADJUST, 305  
 COORDINATE\_METRICS\_CALCULATE,  
 305  
 COORDINATE\_SYSTEM\_CREATE\_-  
   FINISH, 306  
 COORDINATE\_SYSTEM\_CREATE\_-  
   START, 306  
 COORDINATE\_SYSTEM\_DESTROY\_-  
   NUMBER, 306  
 COORDINATE\_SYSTEM\_DESTROY\_PTR,  
 307  
 COORDINATE\_SYSTEM\_DIMENSION\_-  
   GET, 307  
 COORDINATE\_SYSTEM\_DIMENSION\_-  
   SET\_NUMBER, 307  
 COORDINATE\_SYSTEM\_DIMENSION\_-  
   SET\_PTR, 307  
 COORDINATE\_SYSTEM\_FOCUS\_GET,  
 307  
 COORDINATE\_SYSTEM\_FOCUS\_SET\_-  
   NUMBER, 308  
 COORDINATE\_SYSTEM\_FOCUS\_SET\_-  
   PTR, 308  
 COORDINATE\_SYSTEM\_NORMAL\_-  
   CALCULATE, 308  
 COORDINATE\_SYSTEM\_ORIENTATION\_-  
   SET\_NUMBER, 309  
 COORDINATE\_SYSTEM\_ORIENTATION\_-  
   SET\_PTR, 309  
 COORDINATE\_SYSTEM\_ORIGIN\_SET\_-  
   NUMBER, 309  
 COORDINATE\_SYSTEM\_ORIGIN\_SET\_-  
   PTR, 309  
 COORDINATE\_SYSTEM\_RADIAL\_-  
   INTERPOLATION\_TYPE\_GET, 310  
 COORDINATE\_SYSTEM\_RADIAL\_-  
   INTERPOLATION\_TYPE\_SET\_-  
   NUMBER, 310  
 COORDINATE\_SYSTEM\_RADIAL\_-  
   INTERPOLATION\_TYPE\_SET\_PTR,  
 310  
 COORDINATE\_SYSTEM\_TYPE\_GET, 310

COORDINATE\_SYSTEM\_TYPE\_SET\_-  
NUMBER, 310  
COORDINATE\_SYSTEM\_TYPE\_SET\_PTR,  
311  
COORDINATE\_SYSTEM\_USER\_-  
NUMBER\_FIND, 311  
COORDINATE\_SYSTEMS, 313  
COORDINATE\_SYSTEMS\_FINALISE, 311  
COORDINATE\_SYSTEMS\_INITIALISE,  
312  
D2ZX\_DP, 312  
DXZ\_DP, 312  
DZX\_DP, 313  
GLOBAL\_COORDINATE\_SYSTEM, 313  
COORDINATE\_ROUTINES::COORDINATE\_-  
CONVERT\_FROM\_RC, 828  
COORDINATE\_CONVERT\_FROM\_RC\_DP,  
828  
COORDINATE\_CONVERT\_FROM\_RC\_SP,  
828  
COORDINATE\_ROUTINES::COORDINATE\_-  
CONVERT\_TO\_RC, 829  
COORDINATE\_CONVERT\_TO\_RC\_DP,  
829  
COORDINATE\_CONVERT\_TO\_RC\_SP, 829  
COORDINATE\_ROUTINES::COORDINATE\_-  
DELTA\_CALCULATE, 830  
COORDINATE\_DELTA\_CALCULATE\_DP,  
830  
COORDINATE\_ROUTINES::COORDINATE\_-  
DERIVATIVE\_CONVERT\_TO\_RC,  
831  
COORDINATE\_DERIVATIVE\_CONVERT\_-  
TO\_RC\_DP, 831  
COORDINATE\_DERIVATIVE\_CONVERT\_-  
TO\_RC\_SP, 831  
COORDINATE\_ROUTINES::COORDINATE\_-  
SYSTEM\_DESTROY, 832  
COORDINATE\_SYSTEM\_DESTROY\_-  
NUMBER, 832  
COORDINATE\_SYSTEM\_DESTROY\_PTR,  
832  
COORDINATE\_ROUTINES::COORDINATE\_-  
SYSTEM\_DIMENSION\_SET, 833  
COORDINATE\_SYSTEM\_DIMENSION\_-  
SET\_NUMBER, 833  
COORDINATE\_SYSTEM\_DIMENSION\_-  
SET\_PTR, 833  
COORDINATE\_ROUTINES::COORDINATE\_-  
SYSTEM\_FOCUS\_SET, 834  
COORDINATE\_SYSTEM\_FOCUS\_SET\_-  
NUMBER, 834  
COORDINATE\_SYSTEM\_FOCUS\_SET\_-  
PTR, 834  
COORDINATE\_ROUTINES::COORDINATE\_-  
SYSTEM\_ORIENTATION\_SET, 835  
COORDINATE\_SYSTEM\_ORIENTATION\_-  
SET\_NUMBER, 835  
COORDINATE\_SYSTEM\_ORIENTATION\_-  
SET\_PTR, 835  
COORDINATE\_ROUTINES::COORDINATE\_-  
SYSTEM\_ORIGIN\_SET, 836  
COORDINATE\_SYSTEM\_ORIGIN\_SET\_-  
NUMBER, 836  
COORDINATE\_SYSTEM\_ORIGIN\_SET\_-  
PTR, 836  
COORDINATE\_ROUTINES::COORDINATE\_-  
SYSTEM\_PTR\_TYPE, 837  
PTR, 837  
COORDINATE\_ROUTINES::COORDINATE\_-  
SYSTEM\_RADIAL\_-  
INTERPOLATION\_TYPE\_SET, 838  
COORDINATE\_SYSTEM\_RADIAL\_-  
INTERPOLATION\_TYPE\_SET\_-  
NUMBER, 838  
COORDINATE\_SYSTEM\_RADIAL\_-  
INTERPOLATION\_TYPE\_SET\_PTR,  
838  
COORDINATE\_ROUTINES::COORDINATE\_-  
SYSTEM\_TYPE\_SET, 839  
COORDINATE\_SYSTEM\_TYPE\_SET\_-  
NUMBER, 839  
COORDINATE\_SYSTEM\_TYPE\_SET\_PTR,  
839  
COORDINATE\_ROUTINES::COORDINATE\_-  
SYSTEMS\_TYPE, 840  
COORDINATE\_SYSTEMS, 840  
NUMBER\_OF\_COORDINATE\_SYSTEMS,  
840  
COORDINATE\_ROUTINES::CoordinateSystemTypes,  
30  
COORDINATE\_ROUTINES::D2ZX, 841  
D2ZX\_DP, 841  
COORDINATE\_ROUTINES::DXZ, 842  
DXZ\_DP, 842  
COORDINATE\_ROUTINES::DZX, 843  
DZX\_DP, 843  
COORDINATE\_ROUTINES::JacobianType, 36  
COORDINATE\_ROUTINES::RadialInterpolations,  
34  
COORDINATE\_ROUTINES\_JacobianType  
COORDINATE\_JACOBIAN\_AREA\_TYPE,  
36  
COORDINATE\_JACOBIAN\_LINE\_TYPE,  
36  
COORDINATE\_JACOBIAN\_VOLUME\_-  
TYPE, 36  
COORDINATE\_ROUTINES\_RadialInterpolations

COORDINATE\_NO\_RADIAL\_-  
     INTERPOLATION\_TYPE, 34  
 COORDINATE\_RADIAL\_CUBED\_-  
     INTERPOLATION\_TYPE, 34  
 COORDINATE\_RADIAL\_-  
     INTERPOLATION\_TYPE, 35  
 COORDINATE\_RADIAL\_SQUARED\_-  
     INTERPOLATION\_TYPE, 35  
 COORDINATE\_SPHERICAL\_POLAR\_TYPE  
     COORINDATE\_ROUTINES\_-  
         CoordinateSystemTypes, 32  
 COORDINATE\_SYSTEM  
     TYPES::REGION\_TYPE, 1301  
 COORDINATE\_SYSTEM\_CREATE\_FINISH  
     COORDINATE\_ROUTINES, 306  
 COORDINATE\_SYSTEM\_CREATE\_START  
     COORDINATE\_ROUTINES, 306  
 COORDINATE\_SYSTEM\_DESTROY\_NUMBER  
     COORDINATE\_ROUTINES, 306  
     COORDINATE\_-  
         ROUTINES::COORDINATE\_-  
             SYSTEM\_DESTROY, 832  
 COORDINATE\_SYSTEM\_DESTROY\_PTR  
     COORDINATE\_ROUTINES, 307  
     COORDINATE\_-  
         ROUTINES::COORDINATE\_-  
             SYSTEM\_DESTROY, 832  
 COORDINATE\_SYSTEM\_DIMENSION\_GET  
     COORDINATE\_ROUTINES, 307  
 COORDINATE\_SYSTEM\_DIMENSION\_SET\_-  
     NUMBER  
     COORDINATE\_ROUTINES, 307  
     COORDINATE\_-  
         ROUTINES::COORDINATE\_-  
             SYSTEM\_DIMENSION\_SET, 833  
 COORDINATE\_SYSTEM\_DIMENSION\_SET\_-  
     PTR  
     COORDINATE\_ROUTINES, 307  
     COORDINATE\_-  
         ROUTINES::COORDINATE\_-  
             SYSTEM\_DIMENSION\_SET, 833  
 COORDINATE\_SYSTEM\_FINISHED  
     TYPES::COORDINATE\_SYSTEM\_TYPE,  
         1053  
 COORDINATE\_SYSTEM\_FOCUS\_GET  
     COORDINATE\_ROUTINES, 307  
 COORDINATE\_SYSTEM\_FOCUS\_SET\_-  
     NUMBER  
     COORDINATE\_ROUTINES, 308  
     COORDINATE\_-  
         ROUTINES::COORDINATE\_-  
             SYSTEM\_FOCUS\_SET, 834  
 COORDINATE\_SYSTEM\_FOCUS\_SET\_PTR  
     COORDINATE\_ROUTINES, 308

COORDINATE\_-  
     ROUTINES::COORDINATE\_-  
         SYSTEM\_FOCUS\_SET, 834  
 COORDINATE\_SYSTEM\_NORMAL\_-  
     CALCULATE  
     COORDINATE\_ROUTINES, 308  
 COORDINATE\_SYSTEM\_ORIENTATION\_-  
     SET\_NUMBER  
     COORDINATE\_ROUTINES, 309  
     COORDINATE\_-  
         ROUTINES::COORDINATE\_-  
             SYSTEM\_ORIENTATION\_SET, 835  
 COORDINATE\_SYSTEM\_ORIENTATION\_-  
     SET\_PTR  
     COORDINATE\_ROUTINES, 309  
     COORDINATE\_-  
         ROUTINES::COORDINATE\_-  
             SYSTEM\_ORIENTATION\_SET, 835  
 COORDINATE\_SYSTEM\_ORIGIN\_SET\_-  
     NUMBER  
     COORDINATE\_ROUTINES, 309  
     COORDINATE\_-  
         ROUTINES::COORDINATE\_-  
             SYSTEM\_ORIGIN\_SET, 836  
 COORDINATE\_SYSTEM\_ORIGIN\_SET\_PTR  
     COORDINATE\_ROUTINES, 309  
     COORDINATE\_-  
         ROUTINES::COORDINATE\_-  
             SYSTEM\_ORIGIN\_SET, 836  
 COORDINATE\_SYSTEM\_RADIAL\_-  
     INTERPOLATION\_TYPE\_GET  
     COORDINATE\_ROUTINES, 310  
 COORDINATE\_SYSTEM\_RADIAL\_-  
     INTERPOLATION\_TYPE\_SET\_-  
     NUMBER  
     COORDINATE\_ROUTINES, 310  
     COORDINATE\_-  
         ROUTINES::COORDINATE\_-  
             SYSTEM\_RADIAL\_-  
                 INTERPOLATION\_TYPE\_SET, 838  
 COORDINATE\_SYSTEM\_RADIAL\_-  
     INTERPOLATION\_TYPE\_SET\_PTR  
     COORDINATE\_ROUTINES, 310  
     COORDINATE\_-  
         ROUTINES::COORDINATE\_-  
             SYSTEM\_RADIAL\_-  
                 INTERPOLATION\_TYPE\_SET, 838  
 COORDINATE\_SYSTEM\_TYPE\_GET  
     COORDINATE\_ROUTINES, 310  
 COORDINATE\_SYSTEM\_TYPE\_SET\_-  
     NUMBER  
     COORDINATE\_ROUTINES, 310  
     COORDINATE\_-  
         ROUTINES::COORDINATE\_-

SYSTEM\_TYPE\_SET, 839  
COORDINATE\_SYSTEM\_TYPE\_SET\_PTR  
  COORDINATE\_ROUTINES, 311  
  COORDINATE\_-  
    ROUTINES::COORDINATE\_-  
      SYSTEM\_TYPE\_SET, 839  
COORDINATE\_SYSTEM\_USER\_NUMBER\_-  
  FIND  
  COORDINATE\_ROUTINES, 311  
COORDINATE\_SYSTEMS  
  COORDINATE\_ROUTINES, 313  
  COORDINATE\_-  
    ROUTINES::COORDINATE\_-  
      SYSTEMS\_TYPE, 840  
COORDINATE\_SYSTEMS\_FINALISE  
  COORDINATE\_ROUTINES, 311  
COORDINATE\_SYSTEMS\_INITIALISE  
  COORDINATE\_ROUTINES, 312  
COORINDATE\_ROUTINES\_-  
  CoordinateSystemTypes  
  COORDINATE\_CYCLINDRICAL\_-  
    POLAR\_TYPE, 30  
  COORDINATE\_OBLATE\_SPHEROIDAL\_-  
    TYPE, 31  
  COORDINATE\_PROLATE\_SPHEROIDAL\_-  
    TYPE, 31  
  COORDINATE\_RECTANGULAR\_-  
    CARTESIAN\_TYPE, 32  
  COORDINATE\_SPHERICAL\_POLAR\_-  
    TYPE, 32  
COUPLING\_COEFFICIENTS  
  TYPES::EQUATIONS\_COL\_TO\_SOLVER\_-  
    COLS\_MAP\_TYPE, 1125  
  TYPES::EQUATIONS\_ROW\_TO\_-  
    SOLVER\_ROWS\_MAP\_TYPE, 1168  
  TYPES::JACOBIAN\_COL\_TO\_SOLVER\_-  
    COLS\_MAP\_TYPE, 1243  
  TYPES::SOLVER\_COL\_TO\_EQUATIONS\_-  
    MAP\_TYPE, 1312  
  TYPES::SOLVER\_ROW\_TO\_-  
    EQUATIONS\_SET\_MAP\_TYPE,  
      1328  
  TYPES::VARIABLE\_TO\_SOLVER\_COL\_-  
    MAP\_TYPE, 1339  
CPU\_TIMER  
  TIMER, 746  
CPUTIMER  
  BASE\_ROUTINES::interface, 772  
  TIMER::interface, 1035  
CPUTimer  
  timer\_c.c, 1534  
CREATE\_VALUES\_CACHE  
  TYPES::EQUATIONS\_MAPPING\_TYPE,  
    1150  
  TYPES::SOLUTION\_MAPPING\_TYPE,  
    1306  
CROSS\_PRODUCT\_DP  
  MATHS, 545  
  MATHS::CROSS\_PRODUCT, 960  
CROSS\_PRODUCT\_INTG  
  MATHS, 545  
  MATHS::CROSS\_PRODUCT, 960  
CROSS\_PRODUCT\_SP  
  MATHS, 545  
  MATHS::CROSS\_PRODUCT, 960  
CSTRINGLEN  
  F90C::interface, 844  
CStringLen  
  f90c\_c.c, 1412  
CSTRINGLENGTH  
  F90C, 351  
D2ZX\_DP  
  COORDINATE\_ROUTINES, 312  
  COORDINATE\_ROUTINES::D2ZX, 841  
d:/Users/tyu011/workspace/OpenCMISS-  
  trunk/src/analytic\_analysis\_routines.f90,  
    1347  
d:/Users/tyu011/workspace/OpenCMISS-  
  trunk/src/base\_routines.f90, 1350  
d:/Users/tyu011/workspace/OpenCMISS-  
  trunk/src/basis\_routines.f90, 1356  
d:/Users/tyu011/workspace/OpenCMISS-  
  trunk/src/binary\_file\_c.c, 1357  
d:/Users/tyu011/workspace/OpenCMISS-  
  trunk/src/binary\_file\_f.f90, 1361  
d:/Users/tyu011/workspace/OpenCMISS-  
  trunk/srcblas.f90, 1365  
d:/Users/tyu011/workspace/OpenCMISS-  
  trunk/src/classical\_field\_routines.f90,  
    1367  
d:/Users/tyu011/workspace/OpenCMISS-  
  trunk/src/cmiss.f90, 1369  
d:/Users/tyu011/workspace/OpenCMISS-  
  trunk/src/cmiss\_mpi.f90, 1370  
d:/Users/tyu011/workspace/OpenCMISS-  
  trunk/src/cmiss\_parmetis.f90, 1372  
d:/Users/tyu011/workspace/OpenCMISS-  
  trunk/src/cmiss\_petsc.f90, 1374  
d:/Users/tyu011/workspace/OpenCMISS-  
  trunk/src/cmiss\_petsc\_types.f90, 1383  
d:/Users/tyu011/workspace/OpenCMISS-  
  trunk/src/computational\_-  
    environment.f90, 1385  
d:/Users/tyu011/workspace/OpenCMISS-  
  trunk/src/constants.f90, 1388  
d:/Users/tyu011/workspace/OpenCMISS-  
  trunk/src/coordinate\_routines.f90, 1389

- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/distributed\_matrix\_vector.f90,  
[1394](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/domain\_mappings.f90, [1395](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/elasticity\_routines.f90, [1397](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/electromechanics\_routines.f90,  
[1399](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/equations\_mapping\_-  
routines.f90, [1400](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/equations\_matrices\_-  
routines.f90, [1404](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/equations\_set\_constants.f90,  
[1405](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/equations\_set\_routines.f90,  
[1411](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/f90c\_c.c, [1412](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/f90c\_f.f90, [1413](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/field\_IO\_routines.f90, [1415](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/field\_routines.f90, [1420](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/finite\_elasticity\_routines.f90,  
[1427](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/finite\_element\_routines.f90,  
[1429](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/fluid\_mechanics\_routines.f90,  
[1431](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/generated\_mesh\_routines.f90,  
[1432](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/input\_output.f90, [1436](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/iso\_varying\_string.f90, [1451](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/kinds.f90, [1455](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/lapack.f90, [1457](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/Laplace\_equations\_-  
routines.f90, [1459](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/linear\_elasticity\_routines.f90,  
[1462](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/lists.f90, [1464](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/machine\_constants\_aix.f90,  
[1469](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/machine\_constants\_irix.f90,  
[1471](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/machine\_constants\_linux.f90,  
[1472](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/machine\_constants\_vms.f90,  
[1473](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/machine\_constants\_win32.f90,  
[1474](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/math.f90, [1475](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/matrix\_vector.f90, [1478](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/mesh\_routines.f90, [1490](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/node\_routines.f90, [1498](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/problem\_constants.f90, [1500](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/problem\_routines.f90, [1504](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/region\_routines.f90, [1510](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/solution\_mapping\_routines.f90,  
[1512](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/solver\_matrices\_routines.f90,  
[1513](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/solver\_routines.f90, [1516](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/sorting.f90, [1526](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/strings.f90, [1528](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/timer\_c.c, [1534](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/timer\_f.f90, [1535](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/trees.f90, [1537](#)
- d:/Users/tyu011/workspace/OpenCMISS-  
trunk/src/types.f90, [1540](#)
- D\_CROSS\_PRODUCT\_DP

MATHS, 546  
MATHS::D\_CROSS\_PRODUCT, 961  
D\_CROSS\_PRODUCT\_INTG  
    MATHS, 546  
    MATHS::D\_CROSS\_PRODUCT, 961  
D\_CROSS\_PRODUCT\_SP  
    MATHS, 546  
    MATHS::D\_CROSS\_PRODUCT, 961  
DASUM  
    BLAS::interface, 795  
DATA\_DP  
    TYPES::DISTRIBUTED\_MATRIX\_-  
        PETSC\_TYPE, 1070  
    TYPES::DISTRIBUTED\_VECTOR\_-  
        CMISS\_TYPE, 1077  
    TYPES::MATRIX\_TYPE, 1253  
    TYPES::VECTOR\_TYPE, 1341  
DATA\_INTG  
    TYPES::DISTRIBUTED\_VECTOR\_-  
        CMISS\_TYPE, 1077  
    TYPES::MATRIX\_TYPE, 1253  
    TYPES::VECTOR\_TYPE, 1341  
DATA\_L  
    TYPES::DISTRIBUTED\_VECTOR\_-  
        CMISS\_TYPE, 1077  
    TYPES::MATRIX\_TYPE, 1253  
    TYPES::VECTOR\_TYPE, 1342  
DATA\_SIZE  
    TYPES::DISTRIBUTED\_MATRIX\_-  
        PETSC\_TYPE, 1070  
    TYPES::DISTRIBUTED\_VECTOR\_-  
        CMISS\_TYPE, 1077  
    TYPES::DISTRIBUTED\_VECTOR\_-  
        PETSC\_TYPE, 1079  
DATA\_SP  
    TYPES::DISTRIBUTED\_VECTOR\_-  
        CMISS\_TYPE, 1077  
    TYPES::MATRIX\_TYPE, 1253  
    TYPES::VECTOR\_TYPE, 1342  
DATA\_TYPE  
    LISTS::LIST\_TYPE, 957  
    TYPES::DISTRIBUTED\_MATRIX\_TYPE,  
        1074  
    TYPES::DISTRIBUTED\_VECTOR\_-  
        TRANSFER\_TYPE, 1082  
    TYPES::DISTRIBUTED\_VECTOR\_TYPE,  
        1085  
    TYPES::MATRIX\_TYPE, 1254  
    TYPES::VECTOR\_TYPE, 1342  
DAXPY  
    BLAS::interface, 795  
DCOPY  
    BLAS::interface, 795  
DDOT

BLAS::interface, 795  
DECOMPOSITION  
    TYPES::DECOMPOSITION\_ELEMENTS\_-  
        TYPE, 1057  
    TYPES::DECOMPOSITION\_LINES\_TYPE,  
        1061  
    TYPES::DECOMPOSITION\_TOPOLOGY\_-  
        TYPE, 1063  
    TYPES::DOMAIN\_TYPE, 1119  
    TYPES::FIELD\_TYPE, 1226  
DECOMPOSITION\_ALL\_TYPE  
    MESH\_ROUTINES\_DecompositionTypes,  
        113  
DECOMPOSITION\_CALCULATED\_TYPE  
    MESH\_ROUTINES\_DecompositionTypes,  
        113  
DECOMPOSITION\_CREATE\_FINISH  
    MESH\_ROUTINES, 608  
DECOMPOSITION\_CREATE\_START  
    MESH\_ROUTINES, 609  
DECOMPOSITION\_DESTROY  
    MESH\_ROUTINES, 609  
DECOMPOSITION\_ELEMENT\_DOMAIN\_-  
    CALCULATE  
        MESH\_ROUTINES, 610  
DECOMPOSITION\_ELEMENT\_DOMAIN\_GET  
    MESH\_ROUTINES, 610  
DECOMPOSITION\_ELEMENT\_DOMAIN\_SET  
    MESH\_ROUTINES, 611  
DECOMPOSITION\_FINISHED  
    TYPES::DECOMPOSITION\_TYPE, 1065  
DECOMPOSITION\_MESH\_COMPONENT\_-  
    NUMBER\_GET  
        MESH\_ROUTINES, 611  
DECOMPOSITION\_MESH\_COMPONENT\_-  
    NUMBER\_SET  
        MESH\_ROUTINES, 611  
DECOMPOSITION\_NUMBER\_OF\_DOMAINS\_-  
    GET  
        MESH\_ROUTINES, 612  
DECOMPOSITION\_NUMBER\_OF\_DOMAINS\_-  
    SET  
        MESH\_ROUTINES, 612  
DECOMPOSITION\_TYPE  
    TYPES::DECOMPOSITION\_TYPE, 1065  
DECOMPOSITION\_USER\_DEFINED\_TYPE  
    MESH\_ROUTINES\_DecompositionTypes,  
        114  
DECOMPOSITIONS  
    TYPES::DECOMPOSITION\_TYPE, 1065  
    TYPES::DECOMPOSITIONS\_TYPE, 1067  
    TYPES::MESH\_TYPE, 1269  
DEGENERATE  
    TYPES::BASIS\_TYPE, 1045

DEPENDENT  
   TYPES::EQUATIONS\_SET\_TYPE, 1180

DEPENDENT\_FIELD  
   TYPES::EQUATIONS\_INTERPOLATION\_-  
     TYPE, 1127

  TYPES::EQUATIONS\_SET\_DEPENDENT\_-  
     TYPE, 1170

DEPENDENT\_FINISHED  
   TYPES::EQUATIONS\_SET\_DEPENDENT\_-  
     TYPE, 1170

DEPENDENT\_INTERP\_PARAMETERS  
   TYPES::EQUATIONS\_INTERPOLATION\_-  
     TYPE, 1127

DEPENDENT\_INTERP\_POINT  
   TYPES::EQUATIONS\_INTERPOLATION\_-  
     TYPE, 1127

DEPENDENT\_TYPE  
   TYPES::FIELD\_TYPE, 1226

DERIVATIVE\_NUMBERS\_IN\_LOCAL\_LINE  
   TYPES::BASIS\_TYPE, 1045

DERIVATIVE\_ORDER\_INDEX  
   TYPES::BASIS\_TYPE, 1046

DERIVATIVE\_ORDER\_INDEX\_INV  
   TYPES::BASIS\_TYPE, 1046

DERIVATIVES\_IN\_FACE  
   TYPES::DOMAIN\_FACE\_TYPE, 1096

DERIVATIVES\_IN\_LINE  
   TYPES::DOMAIN\_LINE\_TYPE, 1102

DETERMINANT\_FULL\_DP  
   MATHS, 546  
   MATHS::DETERMINANT, 962

DETERMINANT\_FULL\_INTG  
   MATHS, 547  
   MATHS::DETERMINANT, 962

DETERMINANT\_FULL\_SP  
   MATHS, 547  
   MATHS::DETERMINANT, 962

DGESV  
   LAPACK::interface, 938

DIAG\_ALL\_SUBROUTINES  
   BASE\_ROUTINES, 208

DIAG\_FILE\_OPEN  
   BASE\_ROUTINES, 208

DIAG\_FROM\_SUBROUTINE  
   BASE\_ROUTINES, 208

DIAG\_OR\_TIMING  
   BASE\_ROUTINES, 208

DIAG\_ROUTINE\_LIST  
   BASE\_ROUTINES, 208

DIAGNOSTIC\_OUTPUT\_TYPE  
   BASE\_ROUTINES\_OutputType, 19

DIAGNOSTICS  
   BASE\_ROUTINES, 208

  BASE\_ROUTINES::ROUTINE\_STACK\_-  
     ITEM\_TYPE, 776

DIAGNOSTICS1  
   BASE\_ROUTINES, 208

DIAGNOSTICS2  
   BASE\_ROUTINES, 209

DIAGNOSTICS3  
   BASE\_ROUTINES, 209

DIAGNOSTICS4  
   BASE\_ROUTINES, 209

DIAGNOSTICS5  
   BASE\_ROUTINES, 209

DIAGNOSTICS\_FILE\_UNIT  
   BASE\_ROUTINES\_FileUnits, 23

DIAGNOSTICS\_LEVEL1  
   BASE\_ROUTINES, 209

DIAGNOSTICS\_LEVEL2  
   BASE\_ROUTINES, 210

DIAGNOSTICS\_LEVEL3  
   BASE\_ROUTINES, 210

DIAGNOSTICS\_LEVEL4  
   BASE\_ROUTINES, 210

DIAGNOSTICS\_LEVEL5  
   BASE\_ROUTINES, 210

DIAGNOSTICS\_SET\_OFF  
   BASE\_ROUTINES, 171

DIAGNOSTICS\_SET\_ON  
   BASE\_ROUTINES, 171

DIAGONAL\_NUMBER\_NON\_ZEROS  
   TYPES::DISTRIBUTED\_MATRIX\_-  
     PETSC\_TYPE, 1070

DIMENSION  
   TYPES::FIELD\_TYPE, 1226

DIRECT\_SOLVER  
   TYPES::LINEAR\_SOLVER\_TYPE, 1250

DIRECT\_SOLVER\_TYPE  
   TYPES::LINEAR\_DIRECT\_SOLVER\_-  
     TYPE, 1246

DISPLACEMENTS  
   COMP\_ENVIRONMENT::MPI\_-  
     COMPUTATIONAL\_NODE\_TYPE,  
       826

DISTRIBUTED\_MATRIX  
   TYPES::DISTRIBUTED\_MATRIX\_-  
     CMISS\_TYPE, 1068

  TYPES::DISTRIBUTED\_MATRIX\_-  
     PETSC\_TYPE, 1070

DISTRIBUTED\_VECTOR  
   TYPES::DISTRIBUTED\_VECTOR\_-  
     CMISS\_TYPE, 1077

  TYPES::DISTRIBUTED\_VECTOR\_-  
     PETSC\_TYPE, 1079

DIVERGENCE\_TOLERANCE

TYPES::LINEAR\_ITERATIVE\_SOLVER\_TYPE, 1248  
DNRM2  
    BLAS::interface, 796  
DOF\_INDEX  
    TYPES::DOMAIN\_DOFS\_TYPE, 1089  
    TYPES::DOMAIN\_NODE\_TYPE, 1112  
    TYPES::MESH\_NODE\_TYPE, 1261  
DOF\_LIST  
    TYPES::FIELD\_VARIABLE\_TYPE, 1234  
DOF\_TO\_COLUMNS\_MAP  
    TYPES::VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_MAP\_TYPE, 1335  
DOF\_TO\_COLUMNS\_MAPS  
    TYPES::VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP\_TYPE, 1337  
DOF\_TO\_PARAM\_MAP  
    TYPES::FIELD\_MAPPINGS\_TYPE, 1212  
DOF\_TO\_ROWS\_MAP  
    TYPES::VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_MAP\_TYPE, 1335  
    TYPES::VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP\_TYPE, 1337  
DOF\_TYPE  
    TYPES::FIELD\_DOF\_TO\_PARAM\_MAP\_TYPE, 1200  
DOFS  
    TYPES::DOMAIN\_MAPPINGS\_TYPE, 1109  
    TYPES::DOMAIN\_TOPOLOGY\_TYPE, 1117  
    TYPES::MESH\_TOPOLOGY\_TYPE, 1266  
DOMAIN  
    TYPES::DECOMPOSITION\_TYPE, 1065  
    TYPES::DOMAIN\_DOFS\_TYPE, 1089  
    TYPES::DOMAIN\_ELEMENTS\_TYPE, 1093  
    TYPES::DOMAIN\_FACES\_TYPE, 1098  
    TYPES::DOMAIN\_LINES\_TYPE, 1104  
    TYPES::DOMAIN\_MAPPINGS\_TYPE, 1109  
    TYPES::DOMAIN\_NODES\_TYPE, 1114  
    TYPES::DOMAIN\_TOPOLOGY\_TYPE, 1117  
    TYPES::FIELD\_VARIABLE\_COMPONENT\_TYPE, 1230  
DOMAIN\_LOCAL\_BOUNDARY  
    DOMAIN\_MAPPINGS\_DomainType, 38  
DOMAIN\_LOCAL\_GHOST  
    DOMAIN\_MAPPINGS\_DomainType, 38  
DOMAIN\_LOCAL\_INTERNAL  
    DOMAIN\_MAPPINGS\_DomainType, 38  
DOMAIN\_MAPPING  
    TYPES::DISTRIBUTED\_VECTOR\_TYPE, 1086  
    TYPES::FIELD\_MAPPINGS\_TYPE, 1212  
    TYPES::FIELD\_VARIABLE\_TYPE, 1234  
    DOMAIN\_MAPPINGS\_ADJACENT\_DOMAIN\_FINALISE  
        DOMAIN\_MAPPINGS, 315  
    DOMAIN\_MAPPINGS\_ADJACENT\_DOMAIN\_INITIALISE  
        DOMAIN\_MAPPINGS, 317  
    DOMAIN\_MAPPINGS\_MAPPING\_GLOBAL\_FINALISE  
        DOMAIN\_MAPPINGS, 316  
    DOMAIN\_MAPPINGS\_MAPPING\_GLOBAL\_INITIALISE  
        DOMAIN\_MAPPINGS, 317  
    DOMAIN\_MAPPINGS\_MAPPING\_INITIALISE  
        DOMAIN\_MAPPINGS, 317  
    DOMAIN\_MAPPINGS\_NODES\_FINALISE  
        MESH\_ROUTINES, 613  
    DOMAIN\_MAPPINGS\_NODES\_INITIALISE  
        MESH\_ROUTINES, 613  
DOMAIN\_NUMBER  
    TYPES::DOMAIN\_ADJACENT\_DOMAIN\_TYPE, 1087  
    TYPES::DOMAIN\_GLOBAL\_MAPPING\_TYPE, 1099  
DOMAIN\_TOPOLOGY\_CALCULATE  
    MESH\_ROUTINES, 614  
DOMAIN\_TOPOLOGY\_DOFS\_FINALISE  
    MESH\_ROUTINES, 614  
DOMAIN\_TOPOLOGY\_DOFS\_INITIALISE

MESH\_ROUTINES, 615  
 DOMAIN\_TOPOLOGY\_ELEMENT\_FINALISE  
     MESH\_ROUTINES, 615  
 DOMAIN\_TOPOLOGY\_ELEMENT\_-  
     INITIALISE  
         MESH\_ROUTINES, 616  
 DOMAIN\_TOPOLOGY\_ELEMENTS\_FINALISE  
     MESH\_ROUTINES, 616  
 DOMAIN\_TOPOLOGY\_ELEMENTS\_-  
     INITIALISE  
         MESH\_ROUTINES, 617  
 DOMAIN\_TOPOLOGY\_FINALISE  
     MESH\_ROUTINES, 617  
 DOMAIN\_TOPOLOGY\_INITIALISE  
     MESH\_ROUTINES, 618  
 DOMAIN\_TOPOLOGY\_INITIALISE\_FROM\_-  
     MESH  
         MESH\_ROUTINES, 618  
 DOMAIN\_TOPOLOGY\_LINE\_FINALISE  
     MESH\_ROUTINES, 619  
 DOMAIN\_TOPOLOGY\_LINE\_INITIALISE  
     MESH\_ROUTINES, 619  
 DOMAIN\_TOPOLOGY\_LINES\_FINALISE  
     MESH\_ROUTINES, 619  
 DOMAIN\_TOPOLOGY\_LINES\_INITIALISE  
     MESH\_ROUTINES, 620  
 DOMAIN\_TOPOLOGY\_NODE\_FINALISE  
     MESH\_ROUTINES, 620  
 DOMAIN\_TOPOLOGY\_NODE\_INITIALISE  
     MESH\_ROUTINES, 621  
 DOMAIN\_TOPOLOGY\_NODES\_FINALISE  
     MESH\_ROUTINES, 621  
 DOMAIN\_TOPOLOGY\_NODES\_INITIALISE  
     MESH\_ROUTINES, 622  
 DOMAIN\_TOPOLOGY\_NODES\_-  
     SURROUNDING\_ELEMENTS\_-  
         CALCULATE  
             MESH\_ROUTINES, 622  
 DOUBLE\_COMPLEX\_SIZE  
     MACHINE\_CONSTANTS, 540  
 DOUBLE\_REAL\_SIZE  
     MACHINE\_CONSTANTS, 540  
 DOUBLECOMPLEXTYPE  
     binary\_file\_c.c, 1358  
 DOBLETYPETE  
     binary\_file\_c.c, 1358  
 DP  
     KINDS\_RealKinds, 95  
 DP\_FORMAT  
     BINARY\_FILE::BINARY\_FILE\_INFO\_-  
         TYPE, 780  
 DP\_REAL\_SIZE  
     BINARY\_FILE::BINARY\_FILE\_INFO\_-  
         TYPE, 781  
 DPC  
     KINDS\_ComplexKinds, 99  
 DPC\_REAL\_SIZE  
     BINARY\_FILE::BINARY\_FILE\_INFO\_-  
         TYPE, 781  
 DROT  
     BLAS::interface, 796  
 DROTG  
     BLAS::interface, 796  
 DSCAL  
     BLAS::interface, 796  
 DTRSV  
     BLAS::interface, 796  
 DX\_DXI  
     TYPES::FIELD\_INTERPOLATED\_POINT\_-  
         METRICS\_TYPE, 1205  
 DXI\_RX  
     TYPES::FIELD\_INTERPOLATED\_POINT\_-  
         METRICS\_TYPE, 1205  
 DXZ\_DP  
     COORDINATE\_ROUTINES, 312  
     COORDINATE\_ROUTINES::DXZ, 842  
 DZX\_DP  
     COORDINATE\_ROUTINES, 313  
     COORDINATE\_ROUTINES::DZX, 843  
 ECHO\_FILE\_UNIT  
     BASE\_ROUTINES\_FileUnits, 23  
 ECHO\_OUTPUT  
     BASE\_ROUTINES, 210  
 EDP\_DP  
     MATHS, 547  
     MATHS::EDP, 963  
 EDP\_SP  
     MATHS, 547  
     MATHS::EDP, 963  
 EIGENPROBLEM\_SOLVER  
     TYPES::SOLVER\_TYPE, 1331  
 EIGENVALUE\_FULL\_DP  
     MATHS, 547  
     MATHS::EIGENVALUE, 964  
 EIGENVALUE\_FULL\_SP  
     MATHS, 547  
     MATHS::EIGENVALUE, 964  
 EIGENVECTOR\_FULL\_DP  
     MATHS, 548  
     MATHS::EIGENVECTOR, 965  
 EIGENVECTOR\_FULL\_SP  
     MATHS, 548  
     MATHS::EIGENVECTOR, 965  
 EL  
     ELASTICITY\_ROUTINES, 318  
 ELASTICITY\_EQUATIONS\_SET\_SETUP  
     ELASTICITY\_ROUTINES, 319

ELASTICITYFINITEELEMENT\_-  
CALCULATE  
ELASTICITY\_ROUTINES, 319

ELASTICITYFINITEELEMENT\_-  
JACOBIAN\_EVALUATE  
ELASTICITY\_ROUTINES, 320

ELASTICITYFINITEELEMENT\_-  
RESIDUAL\_EVALUATE  
ELASTICITY\_ROUTINES, 320

ELASTICITYPROBLEM\_CLASS\_TYPE\_SET  
ELASTICITY\_ROUTINES, 321

ELASTICITYPROBLEM\_SETUP  
ELASTICITY\_ROUTINES, 321

ELASTICITY\_ROUTINES, 318  
EL, 318  
ELASTICITY\_EQUATIONS\_SET\_SETUP,  
319

ELASTICITYFINITEELEMENT\_-  
CALCULATE, 319

ELASTICITYFINITEELEMENT\_-  
JACOBIAN\_EVALUATE, 320

ELASTICITYFINITEELEMENT\_-  
RESIDUAL\_EVALUATE, 320

ELASTICITYPROBLEM\_CLASS\_TYPE\_-  
SET, 321

ELASTICITY\_PROBLEM\_SETUP, 321

ELECTROMECHANICS\_ROUTINES, 323

ELEMENT\_DERIVATIVES  
TYPES::DOMAIN\_ELEMENT\_TYPE, 1091

ELEMENT\_DOF2PARAM\_MAP  
TYPES::FIELD\_DOF\_TO\_PARAM\_MAP\_-  
TYPE, 1200

ELEMENT\_DOMAIN  
TYPES::DECOMPOSITION\_TYPE, 1065

ELEMENT\_JACOBIAN  
TYPES::EQUATIONS\_JACOBIAN\_TYPE,  
1133

ELEMENT\_LINES  
TYPES::DECOMPOSITION\_ELEMENT\_-  
TYPE, 1055

TYPES::DECOMPOSITION\_LINE\_TYPE,  
1059

ELEMENT\_MATRIX  
TYPES::EQUATIONS\_MATRIX\_TYPE,  
1165

ELEMENT\_NODES  
TYPES::DOMAIN\_ELEMENT\_TYPE, 1091

ELEMENT\_PARAM2DOF\_MAP  
TYPES::FIELD PARAM\_TO\_DOF\_MAP\_-  
TYPE, 1214

ELEMENT\_PARAMETER\_INDEX  
TYPES::BASIS\_TYPE, 1046

ELEMENT\_RESIDUAL

TYPES::EQUATIONS\_JACOBIAN\_TYPE,  
1133

TYPES::EQUATIONS\_MATRICES\_-  
NONLINEAR\_TYPE, 1153

ELEMENT\_VECTOR  
TYPES::EQUATIONS\_MATRICES\_RHS\_-  
TYPE, 1155

TYPES::EQUATIONS\_MATRICES\_-  
SOURCE\_TYPE, 1157

ELEMENTAL\_INFO\_SET  
FIELD\_IO\_ROUTINES::FIELD\_IO\_-  
ELEMENTALL\_INFO\_SET, 845

ELEMENTS  
TYPES::DECOMPOSITION\_ELEMENTS\_-  
TYPE, 1057

TYPES::DECOMPOSITION\_TOPOLOGY\_-  
TYPE, 1063

TYPES::DOMAIN\_ELEMENTS\_TYPE,  
1093

TYPES::DOMAIN\_MAPPINGS\_TYPE, 1109

TYPES::DOMAIN\_TOPOLOGY\_TYPE,  
1117

TYPES::MESH\_ELEMENTS\_TYPE, 1259

TYPES::MESH\_TOPOLOGY\_TYPE, 1266

ELEMENTS\_FINISHED  
TYPES::MESH\_ELEMENTS\_TYPE, 1259

EMBEDDED\_MESHES  
TYPES::MESH\_TYPE, 1269

EMBEDDING\_MESH  
TYPES::MESH\_TYPE, 1269

ENDIAN\_TYPE  
BINARY\_FILE::BINARY\_FILE\_INFO\_-  
TYPE, 781

ENTERS  
BASE\_ROUTINES, 172

EQUATION\_NUMBER  
TYPES::EQUATIONS\_SET\_ANALYTIC\_-  
TYPE, 1169

EQUATIONS  
TYPES::EQUATIONS\_INTERPOLATION\_-  
TYPE, 1127

TYPES::EQUATIONS\_LINEAR\_DATA\_-  
TYPE, 1136

TYPES::EQUATIONS\_MAPPING\_TYPE,  
1150

TYPES::EQUATIONS\_MATRICES\_TYPE,  
1159

TYPES::EQUATIONS\_NONLINEAR\_-  
DATA\_TYPE, 1167

TYPES::EQUATIONS\_SET\_TO\_SOLVER\_-  
MAP\_TYPE, 1177

TYPES::EQUATIONS\_SET\_TYPE, 1180

TYPES::EQUATIONS\_TIME\_DATA\_TYPE,  
1184

TYPES::SOLVER\_COL\_TO\_EQUATIONS\_-  
     SET\_MAP\_TYPE, 1314  
 EQUATIONS\_COL\_NUMBERS  
     TYPES::SOLVER\_COL\_TO\_EQUATIONS\_-  
         MAP\_TYPE, 1312  
 EQUATIONS\_COL\_SOLVER\_COLS\_MAP  
     TYPES::EQUATIONS\_TO\_SOLVER\_-  
         MAPS\_TYPE, 1186  
 EQUATIONS\_COLUMN\_TO\_DOF\_-  
     VARIABLE\_MAP  
     TYPES::EQUATIONS\_JACOBIAN\_TO\_-  
         VARIABLE\_MAP\_TYPE, 1131  
 EQUATIONS\_FINISHED  
     TYPES::EQUATIONS\_TYPE, 1195  
 EQUATIONS\_MAPPING  
     TYPES::EQUATIONS\_MAPPING\_-  
         LINEAR\_TYPE, 1140  
     TYPES::EQUATIONS\_MAPPING\_-  
         NONLINEAR\_TYPE, 1142  
     TYPES::EQUATIONS\_MAPPING\_RHS\_-  
         TYPE, 1145  
     TYPES::EQUATIONS\_MAPPING\_-  
         SOURCE\_TYPE, 1147  
     TYPES::EQUATIONS\_MATRICES\_TYPE,  
         1159  
     TYPES::EQUATIONS\_TYPE, 1195  
 EQUATIONS\_MAPPING\_CALCULATE  
     EQUATIONS\_MAPPING\_ROUTINES, 326  
 EQUATIONS\_MAPPING\_CREATE\_FINISH  
     EQUATIONS\_MAPPING\_ROUTINES, 327  
 EQUATIONS\_MAPPING\_CREATE\_START  
     EQUATIONS\_MAPPING\_ROUTINES, 327  
 EQUATIONS\_MAPPING\_CREATE\_VALUES\_-  
     CACHE\_FINALISE  
     EQUATIONS\_MAPPING\_ROUTINES, 328  
 EQUATIONS\_MAPPING\_CREATE\_VALUES\_-  
     CACHE\_INITIALISE  
     EQUATIONS\_MAPPING\_ROUTINES, 328  
 EQUATIONS\_MAPPING\_DESTROY  
     EQUATIONS\_MAPPING\_ROUTINES, 329  
 EQUATIONS\_MAPPING\_EQUATIONS\_-  
     JACOBIAN\_TO\_VARIABLE\_MAP\_-  
         FINALISE  
     EQUATIONS\_MAPPING\_ROUTINES, 329  
 EQUATIONS\_MAPPING\_EQUATIONS\_-  
     JACOBIAN\_TO\_VARIABLE\_MAP\_-  
         INITIALISE  
     EQUATIONS\_MAPPING\_ROUTINES, 329  
 EQUATIONS\_MAPPING\_EQUATIONS\_-  
     MATRIX\_TO\_VARIABLE\_MAP\_-  
         FINALISE  
     EQUATIONS\_MAPPING\_ROUTINES, 330  
 EQUATIONS\_MAPPING\_EQUATIONS\_-  
     MATRIX\_TO\_VARIABLE\_MAP\_-  
         INITIALISE

INITIALISE  
 EQUATIONS\_MAPPING\_ROUTINES, 330  
 EQUATIONS\_MAPPING\_FINALISE  
     EQUATIONS\_MAPPING\_ROUTINES, 330  
 EQUATIONS\_MAPPING\_FINISHED  
     TYPES::EQUATIONS\_MAPPING\_TYPE,  
         1150  
 EQUATIONS\_MAPPING\_INITIALISE  
     EQUATIONS\_MAPPING\_ROUTINES, 331  
 EQUATIONS\_MAPPING\_LINEAR\_MAPPING\_-  
     FINALISE  
     EQUATIONS\_MAPPING\_ROUTINES, 331  
 EQUATIONS\_MAPPING\_LINEAR\_MAPPING\_-  
     INITIALISE  
     EQUATIONS\_MAPPING\_ROUTINES, 332  
 EQUATIONS\_MAPPING\_LINEAR\_-  
     MATRICES\_COEFFICIENTS\_SET  
     EQUATIONS\_MAPPING\_ROUTINES, 332  
 EQUATIONS\_MAPPING\_LINEAR\_-  
     MATRICES\_NUMBER\_SET  
     EQUATIONS\_MAPPING\_ROUTINES, 332  
 EQUATIONS\_MAPPING\_LINEAR\_-  
     MATRICES\_VARIABLE\_TYPES\_SET  
     EQUATIONS\_MAPPING\_ROUTINES, 333  
 EQUATIONS\_MAPPING\_NONLINEAR\_-  
     MAPPING\_FINALISE  
     EQUATIONS\_MAPPING\_ROUTINES, 334  
 EQUATIONS\_MAPPING\_NONLINEAR\_-  
     MAPPING\_INITIALISE  
     EQUATIONS\_MAPPING\_ROUTINES, 334  
 EQUATIONS\_MAPPING\_RESIDUAL\_-  
     COEFFICIENT\_SET  
     EQUATIONS\_MAPPING\_ROUTINES, 334  
 EQUATIONS\_MAPPING\_RESIDUAL\_-  
     VARIABLE\_TYPE\_SET  
     EQUATIONS\_MAPPING\_ROUTINES, 335  
 EQUATIONS\_MAPPING\_RHS\_COEFFICIENT\_-  
     SET  
     EQUATIONS\_MAPPING\_ROUTINES, 335  
 EQUATIONS\_MAPPING\_RHS\_MAPPING\_-  
     FINALISE  
     EQUATIONS\_MAPPING\_ROUTINES, 335  
 EQUATIONS\_MAPPING\_RHS\_MAPPING\_-  
     INITIALISE  
     EQUATIONS\_MAPPING\_ROUTINES, 336  
 EQUATIONS\_MAPPING\_RHS\_VARIABLE\_-  
     TYPE\_SET  
     EQUATIONS\_MAPPING\_ROUTINES, 336  
 EQUATIONS\_MAPPING\_ROUTINES, 324  
     EQUATIONS\_MAPPING\_CALCULATE,  
         326  
     EQUATIONS\_MAPPING\_CREATE\_-  
         FINISH, 327

EQUATIONS\_MAPPING\_CREATE\_START,  
  327  
EQUATIONS\_MAPPING\_CREATE\_-  
  VALUES\_CACHE\_FINALISE, 328  
EQUATIONS\_MAPPING\_CREATE\_-  
  VALUES\_CACHE\_INITIALISE, 328  
EQUATIONS\_MAPPING\_DESTROY, 329  
EQUATIONS\_MAPPING\_EQUATIONS\_-  
  JACOBIAN\_TO\_VARIABLE\_MAP\_-  
  FINALISE, 329  
EQUATIONS\_MAPPING\_EQUATIONS\_-  
  JACOBIAN\_TO\_VARIABLE\_MAP\_-  
  INITIALISE, 329  
EQUATIONS\_MAPPING\_EQUATIONS\_-  
  MATRIX\_TO\_VARIABLE\_MAP\_-  
  FINALISE, 330  
EQUATIONS\_MAPPING\_EQUATIONS\_-  
  MATRIX\_TO\_VARIABLE\_MAP\_-  
  INITIALISE, 330  
EQUATIONS\_MAPPING\_FINALISE, 330  
EQUATIONS\_MAPPING\_INITIALISE, 331  
EQUATIONS\_MAPPING\_LINEAR\_-  
  MAPPING\_FINALISE, 331  
EQUATIONS\_MAPPING\_LINEAR\_-  
  MAPPING\_INITIALISE, 332  
EQUATIONS\_MAPPING\_LINEAR\_-  
  MATRICES\_COEFFICIENTS\_SET,  
  332  
EQUATIONS\_MAPPING\_LINEAR\_-  
  MATRICES\_NUMBER\_SET, 332  
EQUATIONS\_MAPPING\_LINEAR\_-  
  MATRICES\_VARIABLE\_TYPES\_SET,  
  333  
EQUATIONS\_MAPPING\_NONLINEAR\_-  
  MAPPING\_FINALISE, 334  
EQUATIONS\_MAPPING\_NONLINEAR\_-  
  MAPPING\_INITIALISE, 334  
EQUATIONS\_MAPPING\_RESIDUAL\_-  
  COEFFICIENT\_SET, 334  
EQUATIONS\_MAPPING\_RESIDUAL\_-  
  VARIABLE\_TYPE\_SET, 335  
EQUATIONS\_MAPPING\_RHS\_-  
  COEFFICIENT\_SET, 335  
EQUATIONS\_MAPPING\_RHS\_-  
  MAPPING\_FINALISE, 335  
EQUATIONS\_MAPPING\_RHS\_-  
  MAPPING\_INITIALISE, 336  
EQUATIONS\_MAPPING\_RHS\_-  
  VARIABLE\_TYPE\_SET, 336  
EQUATIONS\_MAPPING\_SOURCE\_-  
  COEFFICIENT\_SET, 337  
EQUATIONS\_MAPPING\_SOURCE\_-  
  MAPPING\_FINALISE, 337  
EQUATIONS\_MAPPING\_SOURCE\_-  
  MAPPING\_INITIALISE, 338  
EQUATIONS\_MAPPING\_SOURCE\_-  
  TO\_EQUATIONS\_COLUMN\_MAP\_-  
  FINALISE, 338  
EQUATIONS\_MAPPING\_SOURCE\_-  
  TO\_EQUATIONS\_JACOBIAN\_MAP\_-  
  FINALISE, 339  
EQUATIONS\_MAPPING\_SOURCE\_-  
  TO\_EQUATIONS\_JACOBIAN\_MAP\_-  
  INITIALISE, 339  
EQUATIONS\_MAPPING\_SOURCE\_-  
  TO\_EQUATIONS\_MATRICES\_MAP\_-  
  FINALISE, 340  
EQUATIONS\_MAPPING\_SOURCE\_-  
  TO\_EQUATIONS\_MATRICES\_MAP\_-  
  INITIALISE, 340  
EQUATIONS\_MAPPING\_SOURCE\_-  
  COEFFICIENT\_SET  
    EQUATIONS\_MAPPING\_ROUTINES, 337  
EQUATIONS\_MAPPING\_SOURCE\_-  
  MAPPING\_FINALISE  
    EQUATIONS\_MAPPING\_ROUTINES, 337  
EQUATIONS\_MAPPING\_SOURCE\_-  
  MAPPING\_INITIALISE  
    EQUATIONS\_MAPPING\_ROUTINES, 338  
EQUATIONS\_MAPPING\_SOURCE\_-  
  VARIABLE\_TYPE\_SET  
    EQUATIONS\_MAPPING\_ROUTINES, 338  
EQUATIONS\_MAPPING\_VARIABLE\_TO\_-  
  EQUATIONS\_COLUMN\_MAP\_-  
  FINALISE  
    EQUATIONS\_MAPPING\_ROUTINES, 338  
EQUATIONS\_MAPPING\_VARIABLE\_TO\_-  
  EQUATIONS\_JACOBIAN\_MAP\_-  
  FINALISE  
    EQUATIONS\_MAPPING\_ROUTINES, 339  
EQUATIONS\_MAPPING\_VARIABLE\_TO\_-  
  EQUATIONS\_JACOBIAN\_MAP\_-  
  INITIALISE  
    EQUATIONS\_MAPPING\_ROUTINES, 339  
EQUATIONS\_MAPPING\_VARIABLE\_TO\_-  
  EQUATIONS\_MATRICES\_MAP\_-  
  FINALISE  
    EQUATIONS\_MAPPING\_ROUTINES, 340  
EQUATIONS\_MAPPING\_VARIABLE\_TO\_-  
  EQUATIONS\_MATRICES\_MAP\_-  
  INITIALISE  
    EQUATIONS\_MAPPING\_ROUTINES, 340  
EQUATIONS\_MATRICES  
  TYPES::EQUATIONS\_MAPPING\_TYPE,  
  1150

TYPES::EQUATIONS\_MATRICES\_-  
     LINEAR\_TYPE, 1152  
 TYPES::EQUATIONS\_MATRICES\_-  
     NONLINEAR\_TYPE, 1153  
 TYPES::EQUATIONS\_MATRICES\_RHS\_-  
     TYPE, 1155  
 TYPES::EQUATIONS\_MATRICES\_-  
     SOURCE\_TYPE, 1157  
 TYPES::EQUATIONS\_TYPE, 1195  
 EQUATIONS\_MATRICES\_FINISHED  
     TYPES::EQUATIONS\_MATRICES\_TYPE,  
         1160  
 EQUATIONS\_MATRIX  
     TYPES::EQUATIONS\_MATRIX\_TO\_-  
         VARIABLE\_MAP\_TYPE, 1164  
     TYPES::EQUATIONS\_TO\_SOLVER\_-  
         MAPS\_TYPE, 1186  
 EQUATIONS\_MATRIX\_NUMBER  
     TYPES::ELEMENT\_MATRIX\_TYPE, 1122  
     TYPES::EQUATIONS\_TO\_SOLVER\_-  
         MAPS\_TYPE, 1186  
     TYPES::EQUATIONS\_TO\_SOLVER\_-  
         MATRIX\_MAPS\_EM\_TYPE, 1188  
 EQUATIONS\_MATRIX\_NUMBERS  
     TYPES::SOLVER\_COL\_TO\_EQUATIONS\_-  
         MAP\_TYPE, 1312  
     TYPES::VARIABLE\_TO\_EQUATIONS\_-  
         MATRICES\_MAP\_TYPE, 1338  
 EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAPS  
     TYPES::EQUATIONS\_MAPPING\_-  
         LINEAR\_TYPE, 1140  
 EQUATIONS\_ROW\_NUMBER  
     TYPES::SOLVER\_ROW\_TO\_-  
         EQUATIONS\_SET\_MAP\_TYPE,  
             1328  
 EQUATIONS\_ROW\_TO\_RESIDUAL\_DOF\_MAP  
     TYPES::EQUATIONS\_MAPPING\_-  
         NONLINEAR\_TYPE, 1142  
 EQUATIONS\_ROW\_TO\_RHS\_DOF\_MAP  
     TYPES::EQUATIONS\_MAPPING\_RHS\_-  
         TYPE, 1145  
 EQUATIONS\_ROW\_TO\_SOLVER\_ROWS\_-  
     MAPS  
     TYPES::EQUATIONS\_SET\_TO\_SOLVER\_-  
         MAP\_TYPE, 1177  
 EQUATIONS\_ROW\_TO\_SOURCE\_DOF\_MAP  
     TYPES::EQUATIONS\_MAPPING\_-  
         SOURCE\_TYPE, 1147  
 EQUATIONS\_ROW\_TO\_VARIABLE\_DOF\_-  
     MAPS  
     TYPES::EQUATIONS\_MAPPING\_-  
         LINEAR\_TYPE, 1141  
 EQUATIONS\_SET  
     TYPES::EQUATIONS\_SET\_ANALYTIC\_-  
         TYPE, 1169  
     TYPES::EQUATIONS\_SET\_DEPENDENT\_-  
         TYPE, 1170  
     TYPES::EQUATIONS\_SET\_FIXED\_-  
         CONDITIONS\_TYPE, 1171  
     TYPES::EQUATIONS\_SET\_GEOMETRY\_-  
         TYPE, 1173  
     TYPES::EQUATIONS\_SET\_MATERIALS\_-  
         TYPE, 1174  
     TYPES::EQUATIONS\_SET\_SOURCE\_-  
         TYPE, 1176  
     TYPES::EQUATIONS\_TYPE, 1195  
     TYPES::SOLVER\_ROW\_TO\_-  
         EQUATIONS\_SET\_MAP\_TYPE,  
             1328  
 EQUATIONS\_SET\_ADDED\_INDEX  
     TYPES::SOLUTION\_TYPE, 1310  
 EQUATIONS\_SET\_ADVECTION\_DIFFUSION\_-  
     EQUATION\_TYPE  
     EQUATIONS\_SET\_CONSTANTS, 344  
 EQUATIONS\_SET\_BEM SOLUTION\_-  
     METHOD  
     EQUATIONS\_SET\_CONSTANTS\_-  
         SolutionMethods, 54  
 EQUATIONS\_SET Biharmonic\_-  
     EQUATION\_TYPE  
     EQUATIONS\_SET\_CONSTANTS, 344  
 EQUATIONS\_SET\_CLASSICAL\_FIELD\_CLASS  
     EQUATIONS\_SET\_CONSTANTS, 344  
 EQUATIONS\_SET\_CONSTANT\_SOURCE\_-  
     POISSON\_SUBTYPE  
     EQUATIONS\_SET\_CONSTANTS, 344  
 EQUATIONS\_SET\_CONSTANTS, 341  
     EQUATIONS\_SET\_ADVECTION\_-  
         DIFFUSION\_EQUATION\_TYPE,  
             344  
     EQUATIONS\_SET Biharmonic\_-  
         EQUATION\_TYPE, 344  
     EQUATIONS\_SET\_CLASSICAL\_FIELD\_-  
         CLASS, 344  
     EQUATIONS\_SET\_CONSTANT\_-  
         SOURCE\_POISSON\_SUBTYPE,  
             344  
     EQUATIONS\_SET\_DIFFUSION\_-  
         EQUATION\_TYPE, 345  
     EQUATIONS\_SET\_ELASTICITY\_CLASS,  
         345  
     EQUATIONS\_SET\_-  
         ELECTROMAGNETICS\_CLASS,  
             345  
     EQUATIONS\_SET\_ELECTROSTATIC\_-  
         TYPE, 345

EQUATIONS\_SETFINITEELASTICITY\_-  
TYPE, 345  
EQUATIONS\_SETFITTING\_CLASS, 345  
EQUATIONS\_SETFLUID\_MECHANICS\_-  
CLASS, 346  
EQUATIONS\_SETGENERALISED\_-  
LAPLACE\_SUBTYPE, 346  
EQUATIONS\_SETHELMHOLTZ\_-  
EQUATION\_TYPE, 346  
EQUATIONS\_SETLAPLACE\_-  
EQUATION\_TYPE, 346  
EQUATIONS\_SETLINEAR\_ELASTIC\_-  
MODAL\_TYPE, 346  
EQUATIONS\_SETLINEAR\_-  
ELASTICITY\_TYPE, 346  
EQUATIONS\_SETLINEAR\_SOURCE\_-  
POISSON\_SUBTYPE, 347  
EQUATIONS\_SETMAGNETOSTATIC\_-  
TYPE, 347  
EQUATIONS\_SETMAXWELLS\_-  
EQUATIONS\_TYPE, 347  
EQUATIONS\_SETMODAL\_CLASS, 347  
EQUATIONS\_SETNAVIER\_STOKES\_-  
FLUID\_TYPE, 347  
EQUATIONS\_SETNO\_CLASS, 347  
EQUATIONS\_SETNO\_SUBTYPE, 347  
EQUATIONS\_SETNO\_TYPE, 348  
EQUATIONS\_SETOPTIMISATION\_-  
CLASS, 348  
EQUATIONS\_SETPOISSON\_-  
EQUATION\_TYPE, 348  
EQUATIONS\_SETQUADRATIC\_-  
SOURCE\_POISSON\_SUBTYPE,  
348  
EQUATIONS\_SETREACTION\_-  
DIFFUSION\_EQUATION\_TYPE,  
348  
EQUATIONS\_SETSTANDARD\_-  
LAPLACE\_SUBTYPE, 348  
EQUATIONS\_SETSTOKES\_FLUID\_-  
TYPE, 349  
EQUATIONS\_SETWAVE\_EQUATION\_-  
TYPE, 349  
EQUATIONS\_SET\_-  
CONSTANTS::FixedConditions, 46  
EQUATIONS\_SET\_-  
CONSTANTS::JacobianCalculationTypes,  
50  
EQUATIONS\_SET\_-  
CONSTANTS::LinearityTypes, 48  
EQUATIONS\_SETCONSTANTS::OutputTypes,  
57  
EQUATIONS\_SET\_-  
CONSTANTS::SetupActionTypes,

44  
EQUATIONS\_SETCONSTANTS::SetupTypes,  
40  
EQUATIONS\_SET\_-  
CONSTANTS::SolutionMethods, 54  
EQUATIONS\_SETCONSTANTS::SparsityTypes,  
59  
EQUATIONS\_SET\_-  
CONSTANTS::TimeDepedenceTypes,  
52  
EQUATIONS\_SETCONSTANTS\_-  
FixedConditions  
EQUATIONS\_SETFIXED\_BOUNDARY\_-  
CONDITION, 46  
EQUATIONS\_SETMIXED\_BOUNDARY\_-  
CONDITION, 46  
EQUATIONS\_SETNOT\_FIXED, 47  
EQUATIONS\_SETCONSTANTS\_-  
JacobianCalculationTypes  
EQUATIONS\_SETJACOBIAN\_-  
ANALYTIC\_CALCULATED, 50  
EQUATIONS\_SETJACOBIAN\_FD\_-  
CALCULATED, 50  
EQUATIONS\_SETJACOBIAN\_NOT\_-  
CALCULATED, 50  
EQUATIONS\_SETCONSTANTS\_LinearityTypes  
EQUATIONS\_SETLINEAR, 48  
EQUATIONS\_SETNONLINEAR, 48  
EQUATIONS\_SETNONLINEAR\_BCS, 49  
NUMBER\_OF\_EQUATIONS\_SET\_-  
LINEARITY\_TYPES, 49  
EQUATIONS\_SETCONSTANTS\_OutputTypes  
EQUATIONS\_SETELEMENT\_MATRIX\_-  
OUTPUT, 57  
EQUATIONS\_SETMATRIX\_OUTPUT, 57  
EQUATIONS\_SETNO\_OUTPUT, 58  
EQUATIONS\_SETTIMING\_OUTPUT, 58  
EQUATIONS\_SETCONSTANTS\_-  
SetupActionTypes  
EQUATIONS\_SETSETUP\_FINISH\_-  
ACTION, 44  
EQUATIONS\_SETSETUP\_START\_-  
ACTION, 44  
EQUATIONS\_SETCONSTANTS\_SetupTypes  
EQUATIONS\_SETSETUP\_ANALYTIC\_-  
TYPE, 41  
EQUATIONS\_SETSETUP\_DEPENDENT\_-  
TYPE, 41  
EQUATIONS\_SETSETUP\_EQUATIONS\_-  
TYPE, 41  
EQUATIONS\_SETSETUP\_FINAL\_TYPE,  
42  
EQUATIONS\_SETSETUP\_FIXED\_-  
CONDITIONS\_TYPE, 42

EQUATIONS\_SET\_SETUP\_GEOMETRY\_TYPE, 42  
 EQUATIONS\_SET\_SETUP\_INITIAL\_TYPE, 42  
 EQUATIONS\_SET\_SETUP\_MATERIALS\_TYPE, 43  
 EQUATIONS\_SET\_SETUP\_SOURCE\_MATERIALS\_TYPE, 43  
 EQUATIONS\_SET\_SETUP\_SOURCE\_TYPE, 43  
 EQUATIONS\_SET\_CONSTANTS\_SolutionMethods  
 EQUATIONS\_SET\_BEM SOLUTION\_METHOD, 54  
 EQUATIONS\_SET\_FD SOLUTION\_METHOD, 55  
 EQUATIONS\_SET\_FEM SOLUTION\_METHOD, 55  
 EQUATIONS\_SET\_FV SOLUTION\_METHOD, 55  
 EQUATIONS\_SET\_GFEM SOLUTION\_METHOD, 55  
 EQUATIONS\_SET\_GFV SOLUTION\_METHOD, 56  
 NUMBER\_OF\_EQUATIONS\_SET SOLUTION\_METHODS, 56  
 EQUATIONS\_SET\_CONSTANTS\_SparsityTypes  
     EQUATIONS\_SET\_FULL\_MATRICES, 59  
     EQUATIONS\_SET\_SPARSE\_MATRICES, 59  
 EQUATIONS\_SET\_CONSTANTS\_TimeDepedenceTypes  
     EQUATIONS\_SET\_DYNAMIC, 52  
     EQUATIONS\_SET\_QUASISTATIC, 52  
     EQUATIONS\_SET\_STATIC, 53  
     NUMBER\_OF\_EQUATIONS\_SET\_TIME\_TYPES, 53  
 EQUATIONS\_SET\_DIFFUSION\_EQUATION\_TYPE  
     EQUATIONS\_SET\_CONSTANTS, 345  
 EQUATIONS\_SET\_DYNAMIC  
     EQUATIONS\_SET\_CONSTANTS\_TimeDepedenceTypes, 52  
 EQUATIONS\_SET\_ELASTICITY\_CLASS  
     EQUATIONS\_SET\_CONSTANTS, 345  
 EQUATIONS\_SET\_ELECTROMAGNETICS\_CLASS  
     EQUATIONS\_SET\_CONSTANTS, 345  
 EQUATIONS\_SET\_ELECTROSTATIC\_TYPE  
     EQUATIONS\_SET\_CONSTANTS, 345  
 EQUATIONS\_SET\_ELEMENT\_MATRIX\_OUTPUT  
     EQUATIONS\_SET\_CONSTANTS\_OutputTypes, 57  
 EQUATIONS\_SET\_FD SOLUTION\_METHOD  
     EQUATIONS\_SET\_CONSTANTS\_SolutionMethods, 55  
 EQUATIONS\_SET\_FEM SOLUTION\_METHOD  
     EQUATIONS\_SET\_CONSTANTS\_SolutionMethods, 55  
 EQUATIONS\_SET\_FINISHED  
     TYPES::EQUATIONS\_SET\_TYPE, 1180  
 EQUATIONS\_SETFINITE\_ELASTICITY\_TYPE  
     EQUATIONS\_SET\_CONSTANTS, 345  
 EQUATIONS\_SET\_FITTING\_CLASS  
     EQUATIONS\_SET\_CONSTANTS, 345  
 EQUATIONS\_SET\_FIXED\_BOUNDARY\_CONDITION  
     EQUATIONS\_SET\_CONSTANTS\_FixedConditions, 46  
 EQUATIONS\_SET\_FLUID\_MECHANICS\_CLASS  
     EQUATIONS\_SET\_CONSTANTS, 346  
 EQUATIONS\_SET\_FULL\_MATRICES  
     EQUATIONS\_SET\_CONSTANTS\_SparsityTypes, 59  
 EQUATIONS\_SET\_FV SOLUTION\_METHOD  
     EQUATIONS\_SET\_CONSTANTS\_SolutionMethods, 55  
 EQUATIONS\_SET\_GENERALISED\_LAPLACE\_SUBTYPE  
     EQUATIONS\_SET\_CONSTANTS, 346  
 EQUATIONS\_SET\_GFEM SOLUTION\_METHOD  
     EQUATIONS\_SET\_CONSTANTS\_SolutionMethods, 55  
 EQUATIONS\_SET\_GFV SOLUTION\_METHOD  
     EQUATIONS\_SET\_CONSTANTS\_SolutionMethods, 56  
 EQUATIONS\_SET\_HELMHOLTZ\_EQUATION\_TYPE  
     EQUATIONS\_SET\_CONSTANTS, 346  
 EQUATIONS\_SET\_INDEX  
     TYPES::EQUATIONS\_SET\_TO\_SOLVER\_MAP\_TYPE, 1178  
 EQUATIONS\_SET\_INDICES  
     TYPES::SOLVER\_DOF\_TO\_VARIABLE\_MAP\_TYPE, 1318  
 EQUATIONS\_SET\_JACOBIAN\_ANALTYIC\_CALCULATED  
     EQUATIONS\_SET\_CONSTANTS\_JacobianCalculationTypes, 50  
 EQUATIONS\_SET\_JACOBIAN\_FD\_CALCULATED  
     EQUATIONS\_SET\_CONSTANTS\_JacobianCalculationTypes, 50

EQUATIONS\_SET\_JACOBIAN\_NOT\_-  
CALCULATED  
EQUATIONS\_SET\_CONSTANTS\_-  
JacobianCalculationTypes, 50  
EQUATIONS\_SET\_LAPLACE\_EQUATION\_-  
TYPE  
EQUATIONS\_SET\_CONSTANTS, 346  
EQUATIONS\_SET\_LINEAR  
EQUATIONS\_SET\_CONSTANTS\_-  
LinearityTypes, 48  
EQUATIONS\_SET\_LINEAR\_ELASTIC\_-  
MODAL\_TYPE  
EQUATIONS\_SET\_CONSTANTS, 346  
EQUATIONS\_SET\_LINEAR\_ELASTICITY\_-  
TYPE  
EQUATIONS\_SET\_CONSTANTS, 346  
EQUATIONS\_SET\_LINEAR\_SOURCE\_-  
POISSON\_SUBTYPE  
EQUATIONS\_SET\_CONSTANTS, 347  
EQUATIONS\_SET\_MAGNETOSTATIC\_TYPE  
EQUATIONS\_SET\_CONSTANTS, 347  
EQUATIONS\_SET\_MATRIX\_OUTPUT  
EQUATIONS\_SET\_CONSTANTS\_-  
OutputTypes, 57  
EQUATIONS\_SET\_MAXWELLS\_-  
EQUATIONS\_TYPE  
EQUATIONS\_SET\_CONSTANTS, 347  
EQUATIONS\_SET\_MIXED\_BOUNDARY\_-  
CONDITION  
EQUATIONS\_SET\_CONSTANTS\_-  
FixedConditions, 46  
EQUATIONS\_SET\_MODAL\_CLASS  
EQUATIONS\_SET\_CONSTANTS, 347  
EQUATIONS\_SET\_NAVIER\_STOKES\_FLUID\_-  
TYPE  
EQUATIONS\_SET\_CONSTANTS, 347  
EQUATIONS\_SET\_NO\_CLASS  
EQUATIONS\_SET\_CONSTANTS, 347  
EQUATIONS\_SET\_NO\_OUTPUT  
EQUATIONS\_SET\_CONSTANTS\_-  
OutputTypes, 58  
EQUATIONS\_SET\_NO\_SUBTYPE  
EQUATIONS\_SET\_CONSTANTS, 347  
EQUATIONS\_SET\_NO\_TYPE  
EQUATIONS\_SET\_CONSTANTS, 348  
EQUATIONS\_SET\_NONLINEAR  
EQUATIONS\_SET\_CONSTANTS\_-  
LinearityTypes, 48  
EQUATIONS\_SET\_NONLINEAR\_BCS  
EQUATIONS\_SET\_CONSTANTS\_-  
LinearityTypes, 49  
EQUATIONS\_SET\_NOT\_FIXED  
EQUATIONS\_SET\_CONSTANTS\_-  
FixedConditions, 47  
EQUATIONS\_SET\_OPTIMISATION\_CLASS  
EQUATIONS\_SET\_CONSTANTS, 348  
EQUATIONS\_SET\_POISSON\_EQUATION\_-  
TYPE  
EQUATIONS\_SET\_CONSTANTS, 348  
EQUATIONS\_SET\_QUADRATIC\_SOURCE\_-  
POISSON\_SUBTYPE  
EQUATIONS\_SET\_CONSTANTS, 348  
EQUATIONS\_SET\_QUASISTATIC  
EQUATIONS\_SET\_CONSTANTS\_-  
TimeDependenceTypes, 52  
EQUATIONS\_SET\_REACTION\_DIFFUSION\_-  
EQUATION\_TYPE  
EQUATIONS\_SET\_CONSTANTS, 348  
EQUATIONS\_SET\_SETUP\_ANALYTIC\_TYPE  
EQUATIONS\_SET\_CONSTANTS\_-  
SetupTypes, 41  
EQUATIONS\_SET\_SETUP\_DEPENDENT\_-  
TYPE  
EQUATIONS\_SET\_CONSTANTS\_-  
SetupTypes, 41  
EQUATIONS\_SET\_SETUP\_EQUATIONS\_TYPE  
EQUATIONS\_SET\_CONSTANTS\_-  
SetupTypes, 41  
EQUATIONS\_SET\_SETUP\_FINAL\_TYPE  
EQUATIONS\_SET\_CONSTANTS\_-  
SetupTypes, 42  
EQUATIONS\_SET\_SETUP\_FINISH\_ACTION  
EQUATIONS\_SET\_CONSTANTS\_-  
SetupActionTypes, 44  
EQUATIONS\_SET\_SETUP\_FIXED\_-  
CONDITIONS\_TYPE  
EQUATIONS\_SET\_CONSTANTS\_-  
SetupTypes, 42  
EQUATIONS\_SET\_SETUP\_GEOMETRY\_TYPE  
EQUATIONS\_SET\_CONSTANTS\_-  
SetupTypes, 42  
EQUATIONS\_SET\_SETUP\_INITIAL\_TYPE  
EQUATIONS\_SET\_CONSTANTS\_-  
SetupTypes, 42  
EQUATIONS\_SET\_SETUP\_MATERIALS\_TYPE  
EQUATIONS\_SET\_CONSTANTS\_-  
SetupTypes, 43  
EQUATIONS\_SET\_SETUP\_SOURCE\_-  
MATERIALS\_TYPE  
EQUATIONS\_SET\_CONSTANTS\_-  
SetupTypes, 43  
EQUATIONS\_SET\_SETUP\_SOURCE\_TYPE  
EQUATIONS\_SET\_CONSTANTS\_-  
SetupTypes, 43  
EQUATIONS\_SET\_SETUP\_START\_ACTION  
EQUATIONS\_SET\_CONSTANTS\_-  
SetupActionTypes, 44  
EQUATIONS\_SET\_SPARSE\_MATRICES

EQUATIONS\_SET\_CONSTANTS\_-  
     SparsityTypes, 59  
 EQUATIONS\_SET\_STANDARD\_LAPLACE\_-  
     SUBTYPE  
         EQUATIONS\_SET\_CONSTANTS, 348  
 EQUATIONS\_SET\_STATIC  
     EQUATIONS\_SET\_CONSTANTS\_-  
         TimeDepedenceTypes, 53  
 EQUATIONS\_SET\_STOKES\_FLUID\_TYPE  
     EQUATIONS\_SET\_CONSTANTS, 349  
 EQUATIONS\_SET\_TIMING\_OUTPUT  
     EQUATIONS\_SET\_CONSTANTS\_-  
         OutputTypes, 58  
 EQUATIONS\_SET\_TO\_ADD  
     TYPES::SOLUTION\_TYPE, 1310  
 EQUATIONS\_SET\_TO\_SOLVER\_MAP  
     TYPES::SOLUTION\_MAPPING\_TYPE,  
         1306  
 EQUATIONS\_SET\_WAVE\_EQUATION\_TYPE  
     EQUATIONS\_SET\_CONSTANTS, 349  
 EQUATIONS\_SETS  
     TYPES::EQUATIONS\_SET\_TYPE, 1180  
     TYPES::EQUATIONS\_SETS\_TYPE, 1183  
     TYPES::REGION\_TYPE, 1301  
     TYPES::SOLUTION\_MAPPING\_TYPE,  
         1306  
 EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS  
     TYPES::EQUATIONS\_TO\_SOLVER\_-  
         MATRIX\_MAPS\_EM\_TYPE, 1188  
     TYPES::EQUATIONS\_TO\_SOLVER\_-  
         MATRIX\_MAPS\_SM\_TYPE, 1192  
 EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_-  
     EM  
     TYPES::EQUATIONS\_SET\_TO\_SOLVER\_-  
         MAP\_TYPE, 1178  
 EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_-  
     JM  
     TYPES::EQUATIONS\_SET\_TO\_SOLVER\_-  
         MAP\_TYPE, 1178  
 EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_-  
     SM  
     TYPES::EQUATIONS\_SET\_TO\_SOLVER\_-  
         MAP\_TYPE, 1178  
 ERROR\_OUTPUT\_TYPE  
     BASE\_ROUTINES\_OutputType, 20  
 ERROR\_SEPARATOR\_CONSTANT  
     MACHINE\_CONSTANTS, 541  
 ERRORS  
     BASE\_ROUTINES, 182  
 EXCLUSIVE\_CPU\_TIME  
     BASE\_ROUTINES::ROUTINE\_STACK\_-  
         ITEM\_TYPE, 776  
 EXCLUSIVE\_SYSTEM\_TIME

BASE\_ROUTINES::ROUTINE\_STACK\_-  
     ITEM\_TYPE, 777  
 EXITS  
     BASE\_ROUTINES, 194  
 extract\_CH  
     ISO\_VARYING\_STRING, 500  
     ISO\_VARYING\_STRING::extract, 912  
 extract\_VS  
     ISO\_VARYING\_STRING, 500  
     ISO\_VARYING\_STRING::extract, 912  
 F2CSTRING  
     F90C, 351  
 F90C, 350  
     C2FSTRING, 350  
     CSTRINGLENGTH, 351  
     F2CSTRING, 351  
     FSTRINGLENGTH, 351  
 F90C::interface, 844  
     CSTRINGLEN, 844  
     PACKCHARACTERS, 844  
     UNPACKCHARACTERS, 844  
 f90c\_c.c  
     CStringLen, 1412  
     integer, 1412  
     logical, 1412  
     PackCharacters, 1412  
     UnPackCharacters, 1412  
 FACE\_BASES  
     TYPES::BASIS\_TYPE, 1046  
 FACES  
     TYPES::DOMAIN\_FACES\_TYPE, 1098  
     TYPES::DOMAIN\_TOPOLOGY\_TYPE,  
         1118  
 FAMILY\_NUMBER  
     TYPES::BASIS\_TYPE, 1046  
 FEM\_ELEMENT\_MATRICES\_FINALISE  
     FINITE\_ELEMENT\_ROUTINES, 402  
 FEM\_ELEMENT\_MATRICES\_INITIALISE  
     FINITE\_ELEMENT\_ROUTINES, 402  
 FI  
     FIELD\_ROUTINES, 378  
 FIBRE\_FIELD  
     TYPES::EQUATIONS\_INTERPOLATION\_-  
         TYPE, 1128  
     TYPES::EQUATIONS\_SET\_GEOMETRY\_-  
         TYPE, 1173  
 FIBRE\_INTERP\_PARAMETERS  
     TYPES::EQUATIONS\_INTERPOLATION\_-  
         TYPE, 1128  
 FIBRE\_INTERP\_POINT  
     TYPES::EQUATIONS\_INTERPOLATION\_-  
         TYPE, 1128  
 FIBRE\_INTERP\_POINT\_METRICS

TYPES::EQUATIONS\_INTERPOLATION\_-  
TYPE, 1128

FIE  
FIELD\_IO\_ROUTINES, 356

FIELD  
TYPES::FIELD\_INTERPOLATION\_-  
PARAMETERS\_TYPE, 1210

TYPES::FIELD\_PARAMETER\_SETS\_-  
TYPE, 1219

TYPES::FIELD\_VARIABLE\_-  
COMPONENT\_TYPE, 1230

TYPES::FIELD\_VARIABLE\_TYPE, 1234

FIELD\_ANALYTIC\_SET\_TYPE  
FIELD\_ROUTINES\_ParameterSetTypes, 83

FIELD\_ARC\_LENGTH\_SCALING  
FIELD\_ROUTINES\_ScalingTypes, 86

FIELD\_ARITHMETIC\_MEAN\_SCALING  
FIELD\_ROUTINES\_ScalingTypes, 86

FIELD\_BOUNDARY\_CONDITIONS\_SET\_-  
TYPE  
FIELD\_ROUTINES\_ParameterSetTypes, 83

FIELD\_COMPONENT\_INTER  
FIELD\_ROUTINES, 378

FIELD\_COMPONENT\_INTERPOLATION\_GET  
FIELD\_ROUTINES, 379

FIELD\_COMPONENT\_INTERPOLATION\_-  
SET\_NUMBER  
FIELD\_ROUTINES::FIELD\_-  
COMPONENT\_INTERPOLATION\_-  
SET, 852

FIELD\_COMPONENT\_INTERPOLATION\_-  
SET\_PTR  
FIELD\_ROUTINES::FIELD\_-  
COMPONENT\_INTERPOLATION\_-  
SET, 852

FIELD\_COMPONENT\_MESH\_COM  
FIELD\_ROUTINES, 379

FIELD\_COMPONENT\_MESH\_COMPONENT\_-  
GET  
FIELD\_ROUTINES, 380

FIELD\_COMPONENT\_MESH\_COMPONENT\_-  
SET\_NUMBER  
FIELD\_ROUTINES::FIELD\_-  
COMPONENT\_MESH\_-  
COMPONENT\_SET, 853

FIELD\_COMPONENT\_MESH\_COMPONENT\_-  
SET\_PTR  
FIELD\_ROUTINES::FIELD\_-  
COMPONENT\_MESH\_-  
COMPONENT\_SET, 853

FIELD\_CONSTANT\_DOF\_TYPE  
FIELD\_ROUTINES\_DofTypes, 81

FIELD\_CONSTANT\_INTERPOLATION  
FIELD\_ROUTINES\_InterpolationTypes, 75

FIELD\_CREATE\_FINISH  
FIELD\_ROUTINES, 380

FIELD\_CREATE\_VALUES\_CACHE\_FINALISE  
FIELD\_ROUTINES, 381

FIELD\_CREATE\_VALUES\_CACHE\_-  
INITIALISE  
FIELD\_ROUTINES, 381

FIELD\_DEPENDENT\_TYPE  
FIELD\_ROUTINES\_DependentTypes, 69

FIELD\_DEPENDENT\_TYPE\_GET  
FIELD\_ROUTINES, 382

FIELD\_DEPENDENT\_TYPE\_SET\_NUMBER  
FIELD\_ROUTINES, 382

FIELD\_ROUTINES::FIELD\_-  
DEPENDENT\_TYPE\_SET, 854

FIELD\_DEPENDENT\_TYPE\_SET\_PTR  
FIELD\_ROUTINES, 382

FIELD\_ROUTINES::FIELD\_-  
DEPENDENT\_TYPE\_SET, 854

FIELD\_DESTROY  
FIELD\_ROUTINES, 383

FIELD\_DIMENSION\_GET  
FIELD\_ROUTINES, 383

FIELD\_DIMENSION\_SET\_NUMBER  
FIELD\_ROUTINES, 384

FIELD\_ROUTINES::FIELD\_DIMENSION\_-  
SET, 855

FIELD\_DIMENSION\_SET\_PTR  
FIELD\_ROUTINES, 384

FIELD\_ROUTINES::FIELD\_DIMENSION\_-  
SET, 855

FIELD\_ELEMENT\_BASED\_INTERPOLATION  
FIELD\_ROUTINES\_InterpolationTypes, 75

FIELD\_ELEMENT\_DOF\_TYPE  
FIELD\_ROUTINES\_DofTypes, 81

FIELD\_FIBRE\_TYPE  
FIELD\_ROUTINES\_FieldTypes, 73

FIELD\_FINISHED  
TYPES::FIELD\_TYPE, 1226

FIELD\_GAUSS\_POINT\_BASED\_-  
INTERPOLATION  
FIELD\_ROUTINES\_InterpolationTypes, 76

FIELD\_GENERAL\_TYPE  
FIELD\_ROUTINES\_FieldTypes, 73

FIELD\_GEOMETRIC\_FIELD\_SET\_NUMBER  
FIELD\_ROUTINES::FIELD\_-  
GEOMETRIC\_FIELD\_SET, 856

FIELD\_GEOMETRIC\_FIELD\_SET\_PTR  
FIELD\_ROUTINES::FIELD\_-  
GEOMETRIC\_FIELD\_SET, 856

FIELD\_GEOMETRIC\_TYPE  
FIELD\_ROUTINES\_FieldTypes, 74

FIELD\_GRID\_POINT\_BASED\_-  
INTERPOLATION

FIELD\_ROUTINES\_InterpolationTypes, 76  
 FIELD\_HARMONIC\_MEAN\_SCALING  
     FIELD\_ROUTINES\_ScalingTypes, 87  
 FIELD\_INDEPENDENT\_TYPE  
     FIELD\_ROUTINES\_DependentTypes, 69  
 FIELD\_INITIAL\_CONDITIONS\_SET\_TYPE  
     FIELD\_ROUTINES\_ParameterSetTypes, 84  
 FIELD\_INTERPOLATE\_GAUSS  
     FIELD\_ROUTINES, 385  
 FIELD\_INTERPOLATE\_XI  
     FIELD\_ROUTINES, 385  
 FIELD\_INTERPOLATED\_POINT\_FINALISE  
     FIELD\_ROUTINES, 386  
 FIELD\_INTERPOLATED\_POINT\_INITIALISE  
     FIELD\_ROUTINES, 387  
 FIELD\_INTERPOLATED\_POINT\_METRICS\_-  
     CALCULATE  
     FIELD\_ROUTINES, 387  
 FIELD\_INTERPOLATED\_POINT\_METRICS\_-  
     FINALISE  
     FIELD\_ROUTINES, 388  
 FIELD\_INTERPOLATED\_POINT\_METRICS\_-  
     INITIALISE  
     FIELD\_ROUTINES, 388  
 FIELD\_INTERPOLATION\_PARAMETERS\_-  
     ELEMENT\_GET  
     FIELD\_ROUTINES, 388  
 FIELD\_INTERPOLATION\_PARAMETERS\_-  
     FINALISE  
     FIELD\_ROUTINES, 389  
 FIELD\_INTERPOLATION\_PARAMETERS\_-  
     INITIALISE  
     FIELD\_ROUTINES, 390  
 FIELD\_INTERPOLATION\_PARAMETERS\_-  
     LINE\_GET  
     FIELD\_ROUTINES, 390  
 FIELD\_IO\_BASIS\_LHTP\_FAMILY\_LABEL  
     FIELD\_IO\_ROUTINES, 356  
 FIELD\_IO\_COMPONENT\_LABEL  
     FIELD\_IO\_ROUTINES, 371  
 FIELD\_IO\_CREATE\_FIELDS  
     FIELD\_IO\_ROUTINES, 356  
 FIELD\_IO\_DERIVATIVE\_INFO  
     FIELD\_IO\_ROUTINES, 357  
 FIELD\_IO\_DERIVATIVE\_LABEL  
     FIELD\_IO\_ROUTINES, 371  
 FIELD\_IO\_ELEMENTAL\_INFO\_SET\_-  
     ATTACH\_LOCAL\_PROCESS  
     FIELD\_IO\_ROUTINES, 357  
 FIELD\_IO\_ELEMENTAL\_INFO\_SET\_-  
     FINALIZE  
     FIELD\_IO\_ROUTINES, 357  
 FIELD\_IO\_ELEMENTAL\_INFO\_SET\_-  
     INITIALISE

FIELD\_IO\_ROUTINES, 358  
 FIELD\_IO\_ELEMENTAL\_INFO\_SET\_SORT  
     FIELD\_IO\_ROUTINES, 358  
 FIELD\_IO\_ELEMENTS\_EXPORT  
     FIELD\_IO\_ROUTINES, 358  
 FIELD\_IO\_EXPORT\_ELEMENTAL\_GROUP\_-  
     HEADER\_FORTRAN  
     FIELD\_IO\_ROUTINES, 359  
 FIELD\_IO\_EXPORT\_ELEMENTS\_INTO\_  
     FIELD\_IO\_ROUTINES, 359  
 FIELD\_IO\_EXPORT\_NODAL\_GROUP\_-  
     HEADER\_FORTRA  
     FIELD\_IO\_ROUTINES, 360  
 FIELD\_IO\_EXPORT\_NODES\_INTO\_LOC  
     FIELD\_IO\_ROUTINES, 360  
 FIELD\_IO\_FIELD\_INFO  
     FIELD\_IO\_ROUTINES, 361  
 FIELD\_IO\_FIELD\_LABEL  
     FIELD\_IO\_ROUTINES, 371  
 FIELD\_IO\_FILEDS\_GROUP\_INFO\_GET  
     FIELD\_IO\_ROUTINES, 361  
 FIELD\_IO\_FILEDS\_IMPORT  
     FIELD\_IO\_ROUTINES, 362  
 FIELD\_IO\_FILL\_BASIS\_INFO  
     FIELD\_IO\_ROUTINES, 362  
 FIELD\_IO\_FORTRAN\_FILE\_CLOSE  
     FIELD\_IO\_ROUTINES, 363  
 FIELD\_IO\_FORTRAN\_FILE\_OPEN  
     FIELD\_IO\_ROUTINES, 363  
 FIELD\_IO\_FORTRAN\_FILE\_READ\_DP  
     FIELD\_IO\_ROUTINES, 363  
 FIELD\_IO\_FORTRAN\_FILE\_READ\_INTG  
     FIELD\_IO\_ROUTINES, 364  
 FIELD\_IO\_FORTRAN\_FILE\_READ\_STRING  
     FIELD\_IO\_ROUTINES, 364  
 FIELD\_IO\_FORTRAN\_FILE\_REWIND  
     FIELD\_IO\_ROUTINES, 365  
 FIELD\_IO\_FORTRAN\_FILE\_WRITE\_DP  
     FIELD\_IO\_ROUTINES, 365  
 FIELD\_IO\_FORTRAN\_FILE\_WRITE\_INTG  
     FIELD\_IO\_ROUTINES, 365  
 FIELD\_IO\_FORTRAN\_FILE\_WRITE\_STRING  
     FIELD\_IO\_ROUTINES, 366  
 FIELD\_IO\_IMPORT\_GLOBAL\_MESH  
     FIELD\_IO\_ROUTINES, 366  
 FIELD\_IO\_LABEL\_DERIVATIVE\_INFO\_GET  
     FIELD\_IO\_ROUTINES, 367  
 FIELD\_IO\_LABEL\_FIELD\_INFO\_GET  
     FIELD\_IO\_ROUTINES, 368  
 FIELD\_IO\_MULTI\_FILES\_INFO\_GET  
     FIELD\_IO\_ROUTINES, 368  
 FIELD\_IO\_NODAL\_INFO\_SET\_ATTACH\_-  
     LOCAL\_PROCESS  
     FIELD\_IO\_ROUTINES, 368

FIELD\_IO\_NODAL\_INFO\_SET\_FINALIZE  
FIELD\_IO\_ROUTINES, 369

FIELD\_IO\_NODAL\_INFO\_SET\_INITIALISE  
FIELD\_IO\_ROUTINES, 369

FIELD\_IO\_NODAL\_INFO\_SET\_SORT  
FIELD\_IO\_ROUTINES, 369

FIELD\_IO\_NODES\_EXPORT  
FIELD\_IO\_ROUTINES, 370

FIELD\_IO\_ROUTINES, 352

FIE, 356

FIELD\_IO\_BASIS\_LHTP\_FAMILY\_-  
LABEL, 356

FIELD\_IO\_COMPONENT\_LABEL, 371

FIELD\_IO\_CREATE\_FIELDS, 356

FIELD\_IO\_DERIVATIVE\_INFO, 357

FIELD\_IO\_DERIVATIVE\_LABEL, 371

FIELD\_IO\_ELEMENTAL\_INFO\_SET\_-  
ATTACH\_LOCAL\_PROCESS, 357

FIELD\_IO\_ELEMENTAL\_INFO\_SET\_-  
FINALIZE, 357

FIELD\_IO\_ELEMENTAL\_INFO\_SET\_-  
INITIALISE, 358

FIELD\_IO\_ELEMENTAL\_INFO\_SET\_-  
SORT, 358

FIELD\_IO\_ELEMENTS\_EXPORT, 358

FIELD\_IO\_EXPORT\_ELEMENTAL\_-  
GROUP\_HEADER\_FORTRAN, 359

FIELD\_IO\_EXPORT\_ELEMENTS\_INTO\_,  
359

FIELD\_IO\_EXPORT\_NODAL\_GROUP\_-  
HEADER\_FORTRA, 360

FIELD\_IO\_EXPORT\_NODES\_INTO\_LOC,  
360

FIELD\_IO\_FIELD\_INFO, 361

FIELD\_IO\_FIELD\_LABEL, 371

FIELD\_IO\_FILEDS\_GROUP\_INFO\_GET,  
361

FIELD\_IO\_FILEDS\_IMPORT, 362

FIELD\_IO\_FILL\_BASIS\_INFO, 362

FIELD\_IO\_FORTRAN\_FILE\_CLOSE, 363

FIELD\_IO\_FORTRAN\_FILE\_OPEN, 363

FIELD\_IO\_FORTRAN\_FILE\_READ\_DP,  
363

FIELD\_IO\_FORTRAN\_FILE\_READ\_INTG,  
364

FIELD\_IO\_FORTRAN\_FILE\_READ\_-  
STRING, 364

FIELD\_IO\_FORTRAN\_FILE\_REWIND, 365

FIELD\_IO\_FORTRAN\_FILE\_WRITE\_DP,  
365

FIELD\_IO\_FORTRAN\_FILE\_WRITE\_-  
INTG, 365

FIELD\_IO\_FORTRAN\_FILE\_WRITE\_-  
STRING, 366

FIELD\_IO\_IMPORT\_GLOBAL\_MESH, 366

FIELD\_IO\_LABEL\_DERIVATIVE\_INFO\_-  
GET, 367

FIELD\_IO\_LABEL\_FIELD\_INFO\_GET, 368

FIELD\_IO\_MULTI\_FILES\_INFO\_GET, 368

FIELD\_IO\_NODAL\_INFO\_SET\_ATTACH\_-  
LOCAL\_PROCESS, 368

FIELD\_IO\_NODAL\_INFO\_SET\_FINALIZE,  
369

FIELD\_IO\_NODAL\_INFO\_SET\_-  
INITIALISE, 369

FIELD\_IO\_NODAL\_INFO\_SET\_SORT, 369

FIELD\_IO\_NODES\_EXPORT, 370

FIELD\_IO\_SCALE\_FACTORS\_NUMBER\_-  
TYPE, 372

FIELD\_IO\_SCALE\_FACTORS\_-  
PROPERTY\_TYPE, 372

FIELD\_IO\_TRANSLATE\_LABEL\_INTO\_-  
INTERPOLATION\_TYPE, 370

FIELD\_IO\_VARIABLE\_LABEL, 372

SHAPE\_SIZE, 372

STRING\_TO\_MUTI\_INTEGERS\_VS, 371

STRING\_TO\_MUTI\_REALS\_VS, 371

FIELD\_IO\_ROUTINES::FIELD\_IO\_-  
ELEMENTALL\_INFO\_SET, 845

ELEMENTAL\_INFO\_SET, 845

FIELDS, 845

LIST\_OF\_GLOBAL\_NUMBER, 845

NUMBER\_OF\_ELEMENTS, 846

FIELD\_IO\_ROUTINES::FIELD\_IO\_INFO\_SET,  
847

COMPONENTS, 847

NUMBER\_OF\_COMPONENTS, 847

SAME\_HEADER, 847

FIELD\_IO\_ROUTINES::FIELD\_IO\_NODAL\_-  
INFO\_SET, 848

FIELDS, 848

LIST\_OF\_GLOBAL\_NUMBER, 848

NODAL\_INFO\_SET, 848

NUMBER\_OF\_NODES, 848

FIELD\_IO\_ROUTINES::FIELD\_VARIABLE\_-  
COMPONENT\_PTR\_TYPE, 850

PTR, 850

FIELD\_IO\_ROUTINES::MESH\_ELEMENTS\_-  
TYPE\_PTR\_TYPE, 851

PTR, 851

FIELD\_IO\_SCALE\_FACTORS\_NUMBER\_TYPE

FIELD\_IO\_ROUTINES, 372

FIELD\_IO\_SCALE\_FACTORS\_PROPERTY\_-  
TYPE

FIELD\_IO\_ROUTINES, 372

FIELD\_IO\_TRANSLATE\_LABEL\_INTO\_-  
INTERPOLATION\_TYPE

FIELD\_IO\_ROUTINES, 370

**FIELD\_IO\_VARIABLE\_LABEL**  
 FIELD\_IO\_ROUTINES, 372  
**FIELD\_MATERIAL\_TYPE**  
 FIELD\_ROUTINES\_FieldTypes, 74  
**FIELD\_MESH\_DECOMPOSITION\_SET\_-  
NUMBER**  
 FIELD\_ROUTINES::FIELD\_MESH\_-  
 DECOMPOSITION\_SET, 857  
**FIELD\_MESH\_DECOMPOSITION\_SET\_PTR**  
 FIELD\_ROUTINES::FIELD\_MESH\_-  
 DECOMPOSITION\_SET, 857  
**FIELD\_NO\_SCALING**  
 FIELD\_ROUTINES\_ScalingTypes, 87  
**FIELD\_NODE\_BASED\_INTERPOLATION**  
 FIELD\_ROUTINES\_InterpolationTypes, 76  
**FIELD\_NODE\_DOF\_TYPE**  
 FIELD\_ROUTINES\_DofTypes, 81  
**FIELD\_NORMAL\_VARIABLE\_TYPE**  
 FIELD\_ROUTINES\_VariableTypes, 78  
**FIELD\_NUMBER\_OF\_COMPONENTS\_SET\_-  
NUMBER**  
 FIELD\_ROUTINES::FIELD\_NUMBER\_-  
 OF\_COMPONENTS\_SET, 858  
**FIELD\_NUMBER\_OF\_COMPONENTS\_SET\_-  
PTR**  
 FIELD\_ROUTINES::FIELD\_NUMBER\_-  
 OF\_COMPONENTS\_SET, 858  
**FIELD\_NUMBER\_OF\_SET\_TYPES**  
 FIELD\_ROUTINES\_ParameterSetTypes, 84  
**FIELD\_NUMBER\_OF\_VARIABLE\_TYPES**  
 FIELD\_ROUTINES\_VariableTypes, 79  
**FIELD\_NUMBER\_OF\_VARIABLES\_SET\_-  
NUMBER**  
 FIELD\_ROUTINES::FIELD\_NUMBER\_-  
 OF\_VARIABLES\_SET, 859  
**FIELD\_NUMBER\_OF\_VARIABLES\_SET\_PTR**  
 FIELD\_ROUTINES::FIELD\_NUMBER\_-  
 OF\_VARIABLES\_SET, 859  
**FIELD\_POINT\_DOF\_TYPE**  
 FIELD\_ROUTINES\_DofTypes, 82  
**FIELD\_ROUTINES**, 373  
 FI, 378  
 FIELD\_COMPONENT\_INTER, 378  
 FIELD\_COMPONENT\_INTERPOLATION\_-  
 GET, 379  
 FIELD\_COMPONENT\_MESH\_COM, 379  
 FIELD\_COMPONENT\_MESH\_-  
 COMPONENT\_GET, 380  
 FIELD\_CREATE\_FINISH, 380  
 FIELD\_CREATE\_VALUES\_CACHE\_-  
 FINALISE, 381  
 FIELD\_CREATE\_VALUES\_CACHE\_-  
 INITIALISE, 381  
 FIELD\_DEPENDENT\_TYPE\_GET, 382  
  
**FIELD\_DEPENDENT\_TYPE\_SET\_-  
NUMBER**, 382  
**FIELD\_DEPENDENT\_TYPE\_SET\_PTR**,  
 382  
**FIELD\_DESTROY**, 383  
**FIELD\_DIMENSION\_GET**, 383  
**FIELD\_DIMENSION\_SET\_NUMBER**, 384  
**FIELD\_DIMENSION\_SET\_PTR**, 384  
**FIELD\_INTERPOLATE\_GAUSS**, 385  
**FIELD\_INTERPOLATE\_XI**, 385  
**FIELD\_INTERPOLATED\_POINT\_-  
FINALISE**, 386  
**FIELD\_INTERPOLATED\_POINT\_-  
INITIALISE**, 387  
**FIELD\_INTERPOLATED\_POINT\_-  
METRICS\_CALCULATE**, 387  
**FIELD\_INTERPOLATED\_POINT\_-  
METRICS\_FINALISE**, 388  
**FIELD\_INTERPOLATED\_POINT\_-  
METRICS\_INITIALISE**, 388  
**FIELD\_INTERPOLATION\_-  
PARAMETERS\_ELEMENT\_GET**,  
 388  
**FIELD\_INTERPOLATION\_-  
PARAMETERS\_FINALISE**, 389  
**FIELD\_INTERPOLATION\_-  
PARAMETERS\_INITIALISE**, 390  
**FIELD\_INTERPOLATION\_-  
PARAMETERS\_LINE\_GET**, 390  
**FIELD\_VARIABLE\_COMPONENT\_-  
FINALISE**, 391  
**FIELD\_VARIABLE\_COMPONENT\_-  
INITIALISE**, 391  
**FIELD\_VARIABLES\_FINALISE**, 392  
**FIELD\_VARIABLES\_INITIALISE**, 392  
**FIELDS\_FINALISE**, 393  
**FIELDS\_INITIALISE**, 393  
**FIELD\_ROUTINES::DependentTypes**, 60  
**FIELD\_ROUTINES::DimensionTypes**, 71  
**FIELD\_ROUTINES::DofTypes**, 81  
**FIELD\_ROUTINES::FIELD\_COMPONENT\_-  
INTERPOLATION\_SET**, 852  
**FIELD\_COMPONENT\_INTERPOLATION\_-  
SET\_NUMBER**, 852  
**FIELD\_COMPONENT\_INTERPOLATION\_-  
SET\_PTR**, 852  
**FIELD\_ROUTINES::FIELD\_COMPONENT\_-  
MESH\_COMPONENT\_SET**, 853  
**FIELD\_COMPONENT\_MESH\_-  
COMPONENT\_SET\_NUMBER**, 853  
**FIELD\_COMPONENT\_MESH\_-  
COMPONENT\_SET\_PTR**, 853  
**FIELD\_ROUTINES::FIELD\_DEPENDENT\_-  
TYPE\_SET**, 854

FIELD\_DEPENDENT\_TYPE\_SET\_-  
NUMBER, 854  
FIELD\_DEPENDENT\_TYPE\_SET\_PTR,  
854  
FIELD\_ROUTINES::FIELD\_DIMENSION\_SET,  
855  
FIELD\_DIMENSION\_SET\_NUMBER, 855  
FIELD\_DIMENSION\_SET\_PTR, 855  
FIELD\_ROUTINES::FIELD\_GEOMETRIC\_-  
FIELD\_SET, 856  
FIELD\_GEOMETRIC\_FIELD\_SET\_-  
NUMBER, 856  
FIELD\_GEOMETRIC\_FIELD\_SET\_PTR,  
856  
FIELD\_ROUTINES::FIELD\_MESH\_-  
DECOMPOSITION\_SET, 857  
FIELD\_MESH\_DECOMPOSITION\_SET\_-  
NUMBER, 857  
FIELD\_MESH\_DECOMPOSITION\_SET\_-  
PTR, 857  
FIELD\_ROUTINES::FIELD\_NUMBER\_OF\_-  
COMPONENTS\_SET, 858  
FIELD\_NUMBER\_OF\_COMPONENTS\_-  
SET\_NUMBER, 858  
FIELD\_NUMBER\_OF\_COMPONENTS\_-  
SET\_PTR, 858  
FIELD\_ROUTINES::FIELD\_NUMBER\_OF\_-  
VARIABLES\_SET, 859  
FIELD\_NUMBER\_OF\_VARIABLES\_SET\_-  
NUMBER, 859  
FIELD\_NUMBER\_OF\_VARIABLES\_SET\_-  
PTR, 859  
FIELD\_ROUTINES::FIELD\_SCALING\_TYPE\_-  
SET, 860  
FIELD\_SCALING\_TYPE\_SET\_NUMBER,  
860  
FIELD\_SCALING\_TYPE\_SET\_PTR, 860  
FIELD\_ROUTINES::FIELD\_TYPE\_SET, 861  
FIELD\_TYPE\_SET\_NUMBER, 861  
FIELD\_TYPE\_SET\_PTR, 861  
FIELD\_ROUTINES::FieldTypes, 73  
FIELD\_ROUTINES::InterpolationTypes, 75  
FIELD\_ROUTINES::ParameterSetTypes, 83  
FIELD\_ROUTINES::ScalingTypes, 86  
FIELD\_ROUTINES::VariableTypes, 78  
FIELD\_ROUTINES\_DependentTypes  
FIELD\_DEPENDENT\_TYPE, 69  
FIELD\_INDEPENDENT\_TYPE, 69  
LAPLACE\_EQUATION\_ANALYTIC\_-  
CALCULATE, 62  
LAPLACE\_EQUATION\_ANALYTIC\_-  
PARAMETER\_SET\_UPDATE, 62  
LAPLACE\_EQUATION\_EQUATIONS\_-  
SET\_SETUP, 63  
LAPLACE\_EQUATION\_EQUATIONS\_-  
SET\_STANDARD\_SETUP, 63  
LAPLACE\_EQUATION\_EQUATIONS\_-  
SET\_SUBTYPE\_SET, 64  
LAPLACE\_EQUATIONFINITE\_-  
ELEMENT\_CALCULATE, 65  
LAPLACE\_EQUATION\_FIXED\_-  
CONDITIONS\_, 65  
LAPLACE\_EQUATION\_INTE, 66  
LAPLACE\_EQUATION\_PROBLEM\_-  
SETUP, 67  
LAPLACE\_EQUATION\_PROBLEM\_-  
STANDARD\_SETUP, 67  
LAPLACE\_EQUATION\_PROBLEM\_-  
SUBTYPE\_SET, 68  
LAPLACE\_EQUATION\_THREE\_DIM\_1, 69  
LAPLACE\_EQUATION\_THREE\_DIM\_2, 69  
LAPLACE\_EQUATION\_TWO\_DIM\_1, 69  
LAPLACE\_EQUATION\_TWO\_DIM\_2, 70  
FIELD\_ROUTINES\_DimensionTypes  
FIELD\_SCALAR\_DIMENSION\_TYPE, 71  
FIELD\_TENSOR\_DIMENSION\_TYPE, 71  
FIELD\_VECTOR\_DIMENSION\_TYPE, 71  
FIELD\_ROUTINES\_DofTypes  
FIELD\_CONSTANT\_DOF\_TYPE, 81  
FIELD\_ELEMENT\_DOF\_TYPE, 81  
FIELD\_NODE\_DOF\_TYPE, 81  
FIELD\_POINT\_DOF\_TYPE, 82  
FIELD\_ROUTINES\_FieldTypes  
FIELD\_FIBRE\_TYPE, 73  
FIELD\_GENERAL\_TYPE, 73  
FIELD\_GEOMETRIC\_TYPE, 74  
FIELD\_MATERIAL\_TYPE, 74  
FIELD\_ROUTINES\_InterpolationTypes  
FIELD\_CONSTANT\_INTERPOLATION, 75  
FIELD\_ELEMENT\_BASED\_-  
INTERPOLATION, 75  
FIELD\_GAUSS\_POINT\_BASED\_-  
INTERPOLATION, 76  
FIELD\_GRID\_POINT\_BASED\_-  
INTERPOLATION, 76  
FIELD\_NODE\_BASED\_INTERPOLATION,  
76  
FIELD\_ROUTINES\_ParameterSetTypes  
FIELD\_ANALYTIC\_SET\_TYPE, 83  
FIELD\_BOUNDARY\_CONDITIONS\_SET\_-  
TYPE, 83  
FIELD\_INITIAL\_CONDITIONS\_SET\_-  
TYPE, 84  
FIELD\_NUMBER\_OF\_SET\_TYPES, 84  
FIELD\_VALUES\_SET\_TYPE, 84  
FIELD\_ROUTINES\_ScalingTypes  
FIELD\_ARC\_LENGTH\_SCALING, 86

FIELD\_ARITHMETIC\_MEAN\_SCALING, 86  
 FIELD\_HARMONIC\_MEAN\_SCALING, 87  
 FIELD\_NO\_SCALING, 87  
 FIELD\_UNIT\_SCALING, 87  
 FIELD\_ROUTINES\_VariableTypes  
     FIELD\_NORMAL\_VARIABLE\_TYPE, 78  
     FIELD\_NUMBER\_OF\_VARIABLE\_TYPES, 79  
     FIELD\_STANDARD\_VARIABLE\_TYPE, 79  
     FIELD\_TIME\_DERIV1\_VARIABLE\_TYPE, 79  
     FIELD\_TIME\_DERIV2\_VARIABLE\_TYPE, 80  
 FIELD\_SCALAR\_DIMENSION\_TYPE  
     FIELD\_ROUTINES\_DimensionTypes, 71  
 FIELD\_SCALING\_TYPE\_SET\_NUMBER  
     FIELD\_ROUTINES::FIELD\_SCALING\_-  
         TYPE\_SET, 860  
 FIELD\_SCALING\_TYPE\_SET\_PTR  
     FIELD\_ROUTINES::FIELD\_SCALING\_-  
         TYPE\_SET, 860  
 FIELD\_STANDARD\_VARIABLE\_TYPE  
     FIELD\_ROUTINES\_VariableTypes, 79  
 FIELD\_TENSOR\_DIMENSION\_TYPE  
     FIELD\_ROUTINES\_DimensionTypes, 71  
 FIELD\_TIME\_DERIV1\_VARIABLE\_TYPE  
     FIELD\_ROUTINES\_VariableTypes, 79  
 FIELD\_TIME\_DERIV2\_VARIABLE\_TYPE  
     FIELD\_ROUTINES\_VariableTypes, 80  
 FIELD\_TYPE\_SET\_NUMBER  
     FIELD\_ROUTINES::FIELD\_TYPE\_SET,  
         861  
 FIELD\_TYPE\_SET\_PTR  
     FIELD\_ROUTINES::FIELD\_TYPE\_SET,  
         861  
 FIELD\_UNIT\_SCALING  
     FIELD\_ROUTINES\_ScalingTypes, 87  
 FIELD\_VALUES\_SET\_TYPE  
     FIELD\_ROUTINES\_ParameterSetTypes, 84  
 FIELD\_VARIABLE  
     TYPES::FIELD\_INTERPOLATION\_-  
         PARAMETERS\_TYPE, 1210  
     TYPES::FIELD\_VARIABLE\_-  
         COMPONENT\_TYPE, 1230  
 FIELD\_VARIABLE\_COMPONENT\_FINALISE  
     FIELD\_ROUTINES, 391  
 FIELD\_VARIABLE\_COMPONENT\_INITIALISE  
     FIELD\_ROUTINES, 391  
 FIELD\_VARIABLES\_FINALISE  
     FIELD\_ROUTINES, 392  
 FIELD\_VARIABLES\_INITIALISE  
     FIELD\_ROUTINES, 392  
 FIELD\_VECTOR\_DIMENSION\_TYPE  
     FIELD\_ROUTINES\_DimensionTypes, 71  
 FIELDS  
     FIELD\_IO\_ROUTINES::FIELD\_IO\_-  
         ELEMENTALL\_INFO\_SET, 845  
     FIELD\_IO\_ROUTINES::FIELD\_IO\_-  
         NODAL\_INFO\_SET, 848  
     TYPES::FIELD\_TYPE, 1227  
     TYPES::FIELDS\_TYPE, 1236  
     TYPES::REGION\_TYPE, 1301  
 FIELDS\_FINALISE  
     FIELD\_ROUTINES, 393  
 FIELDS\_INITIALISE  
     FIELD\_ROUTINES, 393  
 FIELDS\_USING  
     TYPES::FIELD\_GEOMETRIC\_-  
         PARAMETERS\_TYPE, 1203  
 FILE\_BEGINNING  
     BINARY\_FILE, 226  
 FILE\_CHANGE\_ENDIAN  
     BINARY\_FILE, 226  
 FILE\_CURRENT  
     BINARY\_FILE, 226  
 FILE\_END  
     BINARY\_FILE, 226  
 FILE\_INFORMATION  
     BINARY\_FILE::BINARY\_FILE\_TYPE, 783  
 FILE\_NAME  
     BINARY\_FILE::BINARY\_FILE\_INFO\_-  
         TYPE, 781  
 FILE\_NUMBER  
     BINARY\_FILE::BINARY\_FILE\_INFO\_-  
         TYPE, 781  
 FILE\_SAME\_ENDIAN  
     BINARY\_FILE, 226  
 FINITE\_ELASTICITY\_EQUATIONS\_SET\_-  
     SETUP  
     FINITE\_ELASTICITY\_ROUTINES, 396  
 FINITE\_ELASTICITY\_EQUATIONS\_SET\_-  
     SUBTYPE\_SET  
     FINITE\_ELASTICITY\_ROUTINES, 397  
 FINITE\_ELASTICITYFINITE\_ELEMENT\_-  
     JACOBIAN\_EVALUATE  
     FINITE\_ELASTICITY\_ROUTINES, 397  
 FINITE\_ELASTICITYFINITE\_ELEMENT\_-  
     RESIDUAL\_EVALUATE  
     FINITE\_ELASTICITY\_ROUTINES, 398  
 FINITE\_ELASTICITY\_GAUSS\_CAUCHY\_-  
     TENSOR  
     FINITE\_ELASTICITY\_ROUTINES, 398  
 FINITE\_ELASTICITY\_GAUSS\_-  
     DEFORMATION\_GRADEINT\_-  
         TENSOR  
     FINITE\_ELASTICITY\_ROUTINES, 399  
 FINITE\_ELASTICITY\_GAUSS\_DFDZ

FINITE\_ELASTICITY\_ROUTINES, 399  
FINITE\_ELASTICITY\_GAUSS\_DXDN  
  FINITE\_ELASTICITY\_ROUTINES, 400  
FINITE\_ELASTICITY\_PROBLEM\_SETUP  
  FINITE\_ELASTICITY\_ROUTINES, 400  
FINITE\_ELASTICITY\_PROBLEM\_SUBTYPE\_SET  
  FINITE\_ELASTICITY\_ROUTINES, 401  
FINITE\_ELASTICITY\_ROUTINES, 395  
  FINITE\_ELASTICITY\_EQUATIONS\_SET\_SETUP, 396  
  FINITE\_ELASTICITY\_EQUATIONS\_SET\_SUBTYPE\_SET, 397  
FINITE\_ELASTICITYFINITE\_ELEMENT\_JACOBIAN\_EVALUATE, 397  
FINITE\_ELASTICITYFINITE\_ELEMENT\_RESIDUAL\_EVALUATE, 398  
FINITE\_ELASTICITY\_GAUSS\_CAUCHY\_TENSOR, 398  
FINITE\_ELASTICITY\_GAUSS\_DEFORMATION\_GRADEINT\_TENSOR, 399  
FINITE\_ELASTICITY\_GAUSS\_DFDZ, 399  
FINITE\_ELASTICITY\_GAUSS\_DXDN,  
  400  
FINITE\_ELASTICITY\_PROBLEM\_SETUP, 400  
FINITE\_ELASTICITY\_PROBLEM\_SUBTYPE\_SET, 401  
FINITE\_ELEMENT\_ROUTINES, 402  
  FEM\_ELEMENT\_MATRICES\_FINALISE, 402  
  FEM\_ELEMENT\_MATRICES\_INITIALISE, 402  
FIXED\_CONDITIONS  
  TYPES::EQUATIONS\_SET\_TYPE, 1181  
FIXED\_CONDITIONS\_FINISHED  
  TYPES::EQUATIONS\_SET\_FIXED\_CONDITIONS\_TYPE, 1171  
FLAG\_ERROR\_C  
  BASE\_ROUTINES, 204  
  BASE\_ROUTINES::FLAG\_ERROR, 769  
FLAG\_ERROR\_VS  
  BASE\_ROUTINES, 204  
  BASE\_ROUTINES::FLAG\_ERROR, 769  
FLAG\_WARNING\_C  
  BASE\_ROUTINES, 205  
  BASE\_ROUTINES::FLAG\_WARNING, 771  
FLAG\_WARNING\_VS  
  BASE\_ROUTINES, 205  
  BASE\_ROUTINES::FLAG\_WARNING, 771  
FLIPENDIAN  
  binary\_file\_c.c, 1358  
  FLOATTYP  
    binary\_file\_c.c, 1358  
  FLUID\_MECHANICS\_ROUTINES, 403  
FOCUS  
  TYPES::COORDINATE\_SYSTEM\_TYPE, 1053  
FROM\_DIAG\_TYPE  
  BASE\_ROUTINES\_DiagnosticTypes, 26  
FROM\_TIMING\_TYPE  
  BASE\_ROUTINES\_TimingTypes, 28  
FSTRINGLENGTH  
  F90C, 351  
GAUSS\_BASIS\_FNS  
  TYPES::QUADRATURE\_SCHEME\_TYPE, 1295  
GAUSS\_ORDER  
  TYPES::QUADRATURE\_TYPE, 1297  
GAUSS\_POSITIONS  
  TYPES::QUADRATURE\_SCHEME\_TYPE, 1295  
GAUSS\_WEIGHTS  
  TYPES::QUADRATURE\_SCHEME\_TYPE, 1296  
GENERAL\_OUTPUT\_TYPE  
  BASE\_ROUTINES\_OutputType, 20  
GENERATED\_MESH  
  TYPES::GENERATED\_MESH\_REGULAR\_TYPE, 1238  
  TYPES::MESH\_TYPE, 1269  
GENERATED\_MESH\_BASIS\_GET  
  GENERATED\_MESH\_ROUTINES, 406  
GENERATED\_MESH\_BASIS\_SET\_NUMBER  
  GENERATED\_MESH\_ROUTINES, 407  
GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_BASIS\_SET, 862  
GENERATED\_MESH\_BASIS\_SET\_PTR  
  GENERATED\_MESH\_ROUTINES, 407  
GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_BASIS\_SET, 862  
GENERATED\_MESH\_CREATE\_FINISH  
  GENERATED\_MESH\_ROUTINES, 407  
GENERATED\_MESH\_CREATE\_START  
  GENERATED\_MESH\_ROUTINES, 408  
GENERATED\_MESH\_DESTROY\_NUMBER  
  GENERATED\_MESH\_ROUTINES, 408  
GENERATED\_MESH\_ROUTINES::GENERATED\_MESH\_DESTROY, 863  
GENERATED\_MESH\_DESTROY\_PTR  
  GENERATED\_MESH\_ROUTINES, 409

GENERATED\_MESH\_-  
   ROUTINES::GENERATED\_MESH\_-  
     DESTROY, 863  
 GENERATED\_MESH\_EXTENT\_GET  
   GENERATED\_MESH\_ROUTINES, 409  
 GENERATED\_MESH\_EXTENT\_SET\_NUMBER  
   GENERATED\_MESH\_ROUTINES, 409  
   GENERATED\_MESH\_-  
     ROUTINES::GENERATED\_MESH\_-  
       EXTENT\_SET, 864  
 GENERATED\_MESH\_EXTENT\_SET\_PTR  
   GENERATED\_MESH\_ROUTINES, 410  
   GENERATED\_MESH\_-  
     ROUTINES::GENERATED\_MESH\_-  
       EXTENT\_SET, 864  
 GENERATED\_MESH\_FINALISE  
   GENERATED\_MESH\_ROUTINES, 410  
 GENERATED\_MESH\_FINISHED  
   TYPES::GENERATED\_MESH\_TYPE, 1240  
 GENERATED\_MESH\_FRACTAL\_TREE\_-  
   MESH\_TYPE  
   GENERATED\_MESH\_ROUTINES\_-  
     GeneratedMeshTypes, 88  
 GENERATED\_MESH\_INITIALISE  
   GENERATED\_MESH\_ROUTINES, 411  
 GENERATED\_MESH\_NUMBER\_OF\_-  
   ELEMENTS\_GET  
   GENERATED\_MESH\_ROUTINES, 411  
 GENERATED\_MESH\_NUMBER\_OF\_-  
   ELEMENTS\_SET\_NUMBER  
   GENERATED\_MESH\_ROUTINES, 411  
 GENERATED\_MESH\_-  
   ROUTINES::GENERATED\_MESH\_-  
     NUMBER\_OF\_ELEMENTS\_SET,  
       865  
 GENERATED\_MESH\_NUMBER\_OF\_-  
   ELEMENTS\_SET\_PTR  
   GENERATED\_MESH\_ROUTINES, 412  
 GENERATED\_MESH\_-  
   ROUTINES::GENERATED\_MESH\_-  
     NUMBER\_OF\_ELEMENTS\_SET,  
       865  
 GENERATED\_MESH\_ORIGIN\_GET  
   GENERATED\_MESH\_ROUTINES, 412  
 GENERATED\_MESH\_ORIGIN\_SET\_NUMBER  
   GENERATED\_MESH\_ROUTINES, 412  
 GENERATED\_MESH\_-  
   ROUTINES::GENERATED\_MESH\_-  
     ORIGIN\_SET, 866  
 GENERATED\_MESH\_ORIGIN\_SET\_PTR  
   GENERATED\_MESH\_ROUTINES, 413  
 GENERATED\_MESH\_-  
   ROUTINES::GENERATED\_MESH\_-  
     ORIGIN\_SET, 866  
 GENERATED\_MESH\_POLAR\_MESH\_TYPE  
   GENERATED\_MESH\_ROUTINES\_-  
     GeneratedMeshTypes, 88  
 GENERATED\_MESH\_REGULAR\_CREATE\_-  
   FINISH  
   GENERATED\_MESH\_ROUTINES, 413  
 GENERATED\_MESH\_REGULAR\_FINALISE  
   GENERATED\_MESH\_ROUTINES, 414  
 GENERATED\_MESH\_REGULAR\_INITIALISE  
   GENERATED\_MESH\_ROUTINES, 414  
 GENERATED\_MESH\_REGULAR\_MESH\_TYPE  
   GENERATED\_MESH\_ROUTINES\_-  
     GeneratedMeshTypes, 88  
 GENERATED\_MESH\_ROUTINES, 404  
   GENERATED\_MESH\_BASIS\_GET, 406  
   GENERATED\_MESH\_BASIS\_SET\_-  
     NUMBER, 407  
   GENERATED\_MESH\_BASIS\_SET\_PTR,  
     407  
   GENERATED\_MESH\_CREATE\_FINISH,  
     407  
   GENERATED\_MESH\_CREATE\_START,  
     408  
   GENERATED\_MESH\_DESTROY\_-  
     NUMBER, 408  
   GENERATED\_MESH\_DESTROY\_PTR, 409  
   GENERATED\_MESH\_EXTENT\_GET, 409  
   GENERATED\_MESH\_EXTENT\_SET\_-  
     NUMBER, 409  
   GENERATED\_MESH\_EXTENT\_SET\_PTR,  
     410  
   GENERATED\_MESH\_FINALISE, 410  
   GENERATED\_MESH\_INITIALISE, 411  
   GENERATED\_MESH\_NUMBER\_OF\_-  
     ELEMENTS\_GET, 411  
   GENERATED\_MESH\_NUMBER\_OF\_-  
     ELEMENTS\_SET\_NUMBER, 411  
   GENERATED\_MESH\_NUMBER\_OF\_-  
     ELEMENTS\_SET\_PTR, 412  
   GENERATED\_MESH\_ORIGIN\_GET, 412  
   GENERATED\_MESH\_ORIGIN\_SET\_-  
     NUMBER, 412  
   GENERATED\_MESH\_ORIGIN\_SET\_PTR,  
     413  
   GENERATED\_MESH\_REGULAR\_-  
     CREATE\_FINISH, 413  
   GENERATED\_MESH\_REGULAR\_-  
     FINALISE, 414  
   GENERATED\_MESH\_REGULAR\_-  
     INITIALISE, 414  
   GENERATED\_MESH\_TYPE\_GET, 414  
   GENERATED\_MESH\_TYPE\_SET\_-  
     NUMBER, 415  
   GENERATED\_MESH\_TYPE\_SET\_PTR, 415

GENERATED\_MESH\_USER\_NUMBER\_-  
FIND, 416  
GENERATED\_MESHES, 417  
GENERATED\_MESHES\_FINALISE, 416  
GENERATED\_MESHES\_INITIALISE, 417  
GENERATED\_MESH\_-  
ROUTINES::GENERATED\_MESH\_-  
BASIS\_SET, 862  
GENERATED\_MESH\_BASIS\_SET\_-  
NUMBER, 862  
GENERATED\_MESH\_BASIS\_SET\_PTR,  
862  
GENERATED\_MESH\_-  
ROUTINES::GENERATED\_MESH\_-  
DESTROY, 863  
GENERATED\_MESH\_DESTROY\_-  
NUMBER, 863  
GENERATED\_MESH\_DESTROY\_PTR, 863  
GENERATED\_MESH\_-  
ROUTINES::GENERATED\_MESH\_-  
EXTENT\_SET, 864  
GENERATED\_MESH\_EXTENT\_SET\_-  
NUMBER, 864  
GENERATED\_MESH\_EXTENT\_SET\_PTR,  
864  
GENERATED\_MESH\_-  
ROUTINES::GENERATED\_MESH\_-  
NUMBER\_OF\_ELEMENTS\_SET,  
865  
GENERATED\_MESH\_NUMBER\_OF\_-  
ELEMENTS\_SET\_NUMBER, 865  
GENERATED\_MESH\_NUMBER\_OF\_-  
ELEMENTS\_SET\_PTR, 865  
GENERATED\_MESH\_-  
ROUTINES::GENERATED\_MESH\_-  
ORIGIN\_SET, 866  
GENERATED\_MESH\_ORIGIN\_SET\_-  
NUMBER, 866  
GENERATED\_MESH\_ORIGIN\_SET\_PTR,  
866  
GENERATED\_MESH\_-  
ROUTINES::GENERATED\_MESH\_-  
TYPE\_SET, 867  
GENERATED\_MESH\_TYPE\_SET\_-  
NUMBER, 867  
GENERATED\_MESH\_TYPE\_SET\_PTR, 867  
GENERATED\_MESH\_-  
ROUTINES::GeneratedMeshTypes,  
88  
GENERATED\_MESH\_ROUTINES\_-  
GeneratedMeshTypes  
GENERATED\_MESH\_FRACTAL\_TREE\_-  
MESH\_TYPE, 88  
GENERATED\_MESH\_POLAR\_MESH\_-  
TYPE, 88  
GENERATED\_MESH\_REGULAR\_MESH\_-  
TYPE, 88  
GENERATED\_MESH\_TYPE\_GET  
GENERATED\_MESH\_ROUTINES, 414  
GENERATED\_MESH\_TYPE\_SET\_NUMBER  
GENERATED\_MESH\_ROUTINES, 415  
GENERATED\_MESH\_-  
ROUTINES::GENERATED\_MESH\_-  
TYPE\_SET, 867  
GENERATED\_MESH\_TYPE\_SET\_PTR  
GENERATED\_MESH\_ROUTINES, 415  
GENERATED\_MESH\_-  
ROUTINES::GENERATED\_MESH\_-  
TYPE\_SET, 867  
GENERATED\_MESH\_USER\_NUMBER\_FIND  
GENERATED\_MESH\_ROUTINES, 416  
GENERATED\_MESHES  
GENERATED\_MESH\_ROUTINES, 417  
TYPES::GENERATED\_MESHES\_TYPE,  
1242  
GENERATED\_MESHES\_FINALISE  
GENERATED\_MESH\_ROUTINES, 416  
GENERATED\_MESHES\_INITIALISE  
GENERATED\_MESH\_ROUTINES, 417  
GENERATED\_TYPE  
TYPES::GENERATED\_MESH\_TYPE, 1240  
GEOMETRIC\_FIELD  
TYPES::EQUATIONS\_INTERPOLATION\_-  
TYPE, 1128  
TYPES::EQUATIONS\_SET\_GEOMETRY\_-  
TYPE, 1173  
TYPES::FIELD\_TYPE, 1227  
GEOMETRIC\_FIELD\_PARAMETERS  
TYPES::FIELD\_TYPE, 1227  
GEOMETRIC\_INTERP\_PARAMETERS  
TYPES::EQUATIONS\_INTERPOLATION\_-  
TYPE, 1128  
GEOMETRIC\_INTERP\_POINT  
TYPES::EQUATIONS\_INTERPOLATION\_-  
TYPE, 1128  
GEOMETRIC\_INTERP\_POINT\_METRICS  
TYPES::EQUATIONS\_INTERPOLATION\_-  
TYPE, 1129  
GEOMETRY  
TYPES::EQUATIONS\_SET\_TYPE, 1181  
get\_  
ISO\_VARYING\_STRING, 500  
ISO\_VARYING\_STRING::get, 913  
GET\_BUFFER\_LEN  
ISO\_VARYING\_STRING, 511  
get\_set\_CH  
ISO\_VARYING\_STRING, 500  
ISO\_VARYING\_STRING::get, 913

get\_set\_VS  
     ISO\_VARYING\_STRING, 500  
     ISO\_VARYING\_STRING::get, 913

get\_unit  
     ISO\_VARYING\_STRING, 500  
     ISO\_VARYING\_STRING::get, 913

get\_unit\_set\_CH  
     ISO\_VARYING\_STRING, 501  
     ISO\_VARYING\_STRING::get, 913

get\_unit\_set\_VS  
     ISO\_VARYING\_STRING, 501  
     ISO\_VARYING\_STRING::get, 913

GHOST\_LIST  
     TYPES::DOMAIN\_MAPPING\_TYPE, 1107

GHOSTING\_TYPE  
     TYPES::DISTRIBUTED\_MATRIX\_TYPE,  
         1074  
     TYPES::DISTRIBUTED\_VECTOR\_TYPE,  
         1086

GL  
     TYPES::FIELD\_INTERPOLATED\_POINT\_-  
         METRICS\_TYPE, 1206

GLOBAL\_BOUNDARY\_CONDITIONS  
     TYPES::EQUATIONS\_SET\_FIXED\_-  
         CONDITIONS\_TYPE, 1171

GLOBAL\_COORDINATE\_SYSTEM  
     COORDINATE\_ROUTINES, 313

GLOBAL\_DOF\_OFFSET  
     TYPES::FIELD\_VARIABLE\_TYPE, 1234

GLOBAL\_ELEMENT\_NODES  
     TYPES::MESH\_ELEMENT\_TYPE, 1258

GLOBAL\_M  
     TYPES::DISTRIBUTED\_MATRIX\_-  
         PETSC\_TYPE, 1071

GLOBAL\_N  
     TYPES::DISTRIBUTED\_MATRIX\_-  
         PETSC\_TYPE, 1071  
     TYPES::DISTRIBUTED\_VECTOR\_-  
         PETSC\_TYPE, 1079

GLOBAL\_NUMBER  
     TYPES::BASIS\_TYPE, 1047  
     TYPES::DECOMPOSITION\_ELEMENT\_-  
         TYPE, 1056  
     TYPES::DECOMPOSITION\_TYPE, 1065  
     TYPES::DOMAIN\_NODE\_TYPE, 1112  
     TYPES::EQUATIONS\_SET\_TYPE, 1181  
     TYPES::FIELD\_TYPE, 1227  
     TYPES::GENERATED\_MESH\_TYPE, 1240  
     TYPES::MESH\_ELEMENT\_TYPE, 1258  
     TYPES::MESH\_NODE\_TYPE, 1261  
     TYPES::MESH\_TYPE, 1269  
     TYPES::NODE\_TYPE, 1273  
     TYPES::PROBLEM\_TYPE, 1291

    TYPES::QUADRATURE\_SCHEME\_TYPE,  
         1296

GLOBAL\_NUMBERS  
     TYPES::DISTRIBUTED\_VECTOR\_-  
         PETSC\_TYPE, 1080

GLOBAL\_REGION  
     REGION\_ROUTINES, 675

GLOBAL\_ROW\_NUMBERS  
     TYPES::DISTRIBUTED\_MATRIX\_-  
         PETSC\_TYPE, 1071

GLOBAL\_TO\_LOCAL\_MAP  
     TYPES::DOMAIN\_MAPPING\_TYPE, 1107

GU  
     TYPES::FIELD\_INTERPOLATED\_POINT\_-  
         METRICS\_TYPE, 1206

HAVE\_JACOBIAN  
     TYPES::SOLUTION\_MAPPING\_TYPE,  
         1306

HEAD  
     BASE\_ROUTINES::ROUTINE\_LIST\_TYPE,  
         775

HEADER  
     BINARY\_FILE::BINARY\_TAG\_TYPE, 784

HEAP\_SORT\_DP  
     SORTING, 724  
     SORTING::HEAP\_SORT, 1015

HEAP\_SORT\_INTG  
     SORTING, 724  
     SORTING::HEAP\_SORT, 1015

HEAP\_SORT\_SP  
     SORTING, 724  
     SORTING::HEAP\_SORT, 1015

HELP\_OUTPUT\_TYPE  
     BASE\_ROUTINES\_OutputType, 20

HERMITE  
     TYPES::BASIS\_TYPE, 1047

I0\_DP  
     MATHS, 548  
     MATHS::I0, 966

I0\_SP  
     MATHS, 548  
     MATHS::I0, 966

I1\_DP  
     MATHS, 548  
     MATHS::I1, 967

I1\_SP  
     MATHS, 548  
     MATHS::I1, 967

iachar\_  
     ISO\_VARYING\_STRING, 501  
     ISO\_VARYING\_STRING::iachar, 915

ichar\_

ISO\_VARYING\_STRING, 501  
ISO\_VARYING\_STRING::ichar, 916

ID  
  TYPES::MATRIX\_TYPE, 1254  
  TYPES::VECTOR\_TYPE, 1342

IN\_DIAG\_TYPE  
  BASE\_ROUTINES\_DiagnosticTypes, 26

IN\_TIMING\_TYPE  
  BASE\_ROUTINES\_TimingTypes, 28

INCLUSIVE\_CPU\_TIME  
  BASE\_ROUTINES::ROUTINE\_STACK\_-  
    ITEM\_TYPE, 777

INCLUSIVE\_SYSTEM\_TIME  
  BASE\_ROUTINES::ROUTINE\_STACK\_-  
    ITEM\_TYPE, 777

INDEX  
  BINARY\_FILE::BINARY\_TAG\_TYPE, 784

index\_CH\_VS  
  ISO\_VARYING\_STRING, 501  
  ISO\_VARYING\_STRING::index, 917

index\_VS\_CH  
  ISO\_VARYING\_STRING, 501  
  ISO\_VARYING\_STRING::index, 917

index\_VS\_VS  
  ISO\_VARYING\_STRING, 501  
  ISO\_VARYING\_STRING::index, 917

INITIAL\_POSITION  
  TYPES::NODE\_TYPE, 1273

INITIAL\_SIZE  
  LISTS::LIST\_TYPE, 958

INPUT\_OUTPUT, 418  
  WR, 432–452  
  WRITE\_STRING\_C, 453  
  WRITE\_STRING\_FMT, 453–457  
  WRITE\_STRING\_FMT\_VALUE\_C, 458  
  WRITE\_STRING\_FMT\_VALUE\_DP, 458  
  WRITE\_STRING\_FMT\_VALUE\_INTG, 459  
  WRITE\_STRING\_FMT\_VALUE\_L, 459  
  WRITE\_STRING\_FMT\_VALUE\_LINTG,  
    460  
  WRITE\_STRING\_FMT\_VALUE\_SP, 460  
  WRITE\_STRING\_FMT\_VALUE\_VS, 461  
  WRITE\_STRING\_IDX, 461–465  
  WRITE\_STRING\_MATRIX\_DP, 465  
  WRITE\_STRING\_MATRIX\_INTG, 467  
  WRITE\_STRING\_MATRIX\_L, 468  
  WRITE\_STRING\_MATRIX\_LINTG, 469  
  WRITE\_STRING\_MATRIX\_SP, 470  
  WRITE\_STRING\_TWO\_VALUE\_C\_C, 471  
  WRITE\_STRING\_TWO\_VALUE\_C\_DP, 472  
  WRITE\_STRING\_TWO\_VALUE\_C\_INTG,  
    473  
  WRITE\_STRING\_TWO\_VALUE\_C\_L, 473  
  WRITE\_STRING\_TWO\_VALUE\_C\_SP, 474

WRITE\_STRING\_TWO\_VALUE\_C\_VS, 474  
WRITE\_STRING\_TWO\_VALUE\_DP\_C, 475  
WRITE\_STRING\_TWO\_VALUE\_DP\_DP,  
  476  
WRITE\_STRING\_TWO\_VALUE\_DP\_INTG,  
  476  
WRITE\_STRING\_TWO\_VALUE\_DP\_L, 477  
WRITE\_STRING\_TWO\_VALUE\_DP\_SP,  
  477  
WRITE\_STRING\_TWO\_VALUE\_DP\_VS,  
  478  
WRITE\_STRING\_TWO\_VALUE\_INTG\_C,  
  478  
WRITE\_STRING\_TWO\_VALUE\_INTG\_DP,  
  479  
WRITE\_STRING\_TWO\_VALUE\_INTG\_-  
  INTG, 480  
WRITE\_STRING\_TWO\_VALUE\_INTG\_L,  
  480  
WRITE\_STRING\_TWO\_VALUE\_INTG\_SP,  
  481  
WRITE\_STRING\_TWO\_VALUE\_INTG\_VS,  
  481  
WRITE\_STRING\_TWO\_VALUE\_L\_C, 482  
WRITE\_STRING\_TWO\_VALUE\_L\_DP, 483  
WRITE\_STRING\_TWO\_VALUE\_L\_INTG,  
  483  
WRITE\_STRING\_TWO\_VALUE\_L\_L, 484  
WRITE\_STRING\_TWO\_VALUE\_L\_SP, 484  
WRITE\_STRING\_TWO\_VALUE\_L\_VS, 485  
WRITE\_STRING\_TWO\_VALUE\_SP\_C, 485  
WRITE\_STRING\_TWO\_VALUE\_SP\_DP,  
  486  
WRITE\_STRING\_TWO\_VALUE\_SP\_INTG,  
  487  
WRITE\_STRING\_TWO\_VALUE\_SP\_L, 487  
WRITE\_STRING\_TWO\_VALUE\_SP\_SP,  
  488  
WRITE\_STRING\_TWO\_VALUE\_SP\_VS,  
  488  
WRITE\_STRING\_TWO\_VALUE\_VS\_C, 489  
WRITE\_STRING\_TWO\_VALUE\_VS\_DP,  
  490  
WRITE\_STRING\_TWO\_VALUE\_VS\_INTG,  
  490  
WRITE\_STRING\_TWO\_VALUE\_VS\_L, 491  
WRITE\_STRING\_TWO\_VALUE\_VS\_SP,  
  491  
WRITE\_STRING\_TWO\_VALUE\_VS\_VS,  
  492  
WRITE\_STRING\_VALUE\_C, 493  
WRITE\_STRING\_VALUE\_DP, 493  
WRITE\_STRING\_VALUE\_INTG, 494  
WRITE\_STRING\_VALUE\_L, 494

WRITE\_STRING\_VALUE\_LINTG, 495  
 WRITE\_STRING\_VALUE\_SP, 495  
 WRITE\_STRING\_VALUE\_VS, 496  
 WRITE\_STRING\_VS, 496  
 INPUT\_OUTPUT::MatrixNameIndexFormat, 90  
 INPUT\_OUTPUT::WRITE\_STRING, 868  
   WRITE\_STRING\_C, 868  
   WRITE\_STRING\_VS, 868  
 INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
   TWO\_VALUE, 869  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_C\_-  
   C, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_C\_-  
   DP, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_C\_-  
   INTG, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_C\_-  
   L, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_C\_-  
   SP, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_C\_-  
   VS, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   DP\_C, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   DP\_DP, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   DP\_INTG, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   DP\_L, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   DP\_SP, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   DP\_VS, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   INTG\_C, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   INTG\_DP, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   INTG\_INTG, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   INTG\_L, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   INTG\_SP, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   INTG\_VS, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_L\_-  
   C, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_L\_-  
   DP, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_L\_-  
   INTG, 871  
   WRITE\_STRING\_FMT\_TWO\_VALUE\_L\_-  
   L, 871  
 WRITE\_STRING\_FMT\_TWO\_VALUE\_L\_-  
   SP, 871  
 WRITE\_STRING\_FMT\_TWO\_VALUE\_L\_-  
   VS, 871  
 WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   SP\_C, 871  
 WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   SP\_DP, 871  
 WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   SP\_INTG, 871  
 WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   SP\_L, 871  
 WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   SP\_SP, 871  
 WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   SP\_VS, 871  
 WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   VS\_C, 871  
 WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   VS\_DP, 871  
 WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   VS\_INTG, 871  
 WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   VS\_L, 871  
 WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   VS\_SP, 871  
 WRITE\_STRING\_FMT\_TWO\_VALUE\_-  
   VS\_VS, 871  
 INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
   VALUE, 872  
   WRITE\_STRING\_FMT\_VALUE\_C, 872  
   WRITE\_STRING\_FMT\_VALUE\_DP, 872  
   WRITE\_STRING\_FMT\_VALUE\_INTG, 873  
   WRITE\_STRING\_FMT\_VALUE\_L, 873  
   WRITE\_STRING\_FMT\_VALUE\_LINTG,  
   874  
   WRITE\_STRING\_FMT\_VALUE\_SP, 874  
   WRITE\_STRING\_FMT\_VALUE\_VS, 874  
 INPUT\_OUTPUT::WRITE\_STRING\_IDX\_-  
   VECTOR, 876  
   WRITE\_STRING\_IDX\_VECTOR\_DP, 876  
   WRITE\_STRING\_IDX\_VECTOR\_INTG,  
   876  
   WRITE\_STRING\_IDX\_VECTOR\_L, 876  
   WRITE\_STRING\_IDX\_VECTOR\_LINTG,  
   876  
   WRITE\_STRING\_IDX\_VECTOR\_SP, 876  
 INPUT\_OUTPUT::WRITE\_STRING\_MATRIX,  
   877  
   WRITE\_STRING\_MATRIX\_DP, 877  
   WRITE\_STRING\_MATRIX\_INTG, 878  
   WRITE\_STRING\_MATRIX\_L, 879  
   WRITE\_STRING\_MATRIX\_LINTG, 880  
   WRITE\_STRING\_MATRIX\_SP, 880

INPUT\_OUTPUT::WRITE\_STRING\_TWO\_-  
  VALUE, 882  
  WRITE\_STRING\_TWO\_VALUE\_C\_C, 883  
  WRITE\_STRING\_TWO\_VALUE\_C\_DP, 884  
  WRITE\_STRING\_TWO\_VALUE\_C\_INTG,  
    884  
  WRITE\_STRING\_TWO\_VALUE\_C\_L, 884  
  WRITE\_STRING\_TWO\_VALUE\_C\_SP, 885  
  WRITE\_STRING\_TWO\_VALUE\_C\_VS, 885  
  WRITE\_STRING\_TWO\_VALUE\_DP\_C, 886  
  WRITE\_STRING\_TWO\_VALUE\_DP\_DP,  
    886  
  WRITE\_STRING\_TWO\_VALUE\_DP\_INTG,  
    887  
  WRITE\_STRING\_TWO\_VALUE\_DP\_L, 887  
  WRITE\_STRING\_TWO\_VALUE\_DP\_SP,  
    888  
  WRITE\_STRING\_TWO\_VALUE\_DP\_VS,  
    888  
  WRITE\_STRING\_TWO\_VALUE\_INTG\_C,  
    889  
  WRITE\_STRING\_TWO\_VALUE\_INTG\_DP,  
    889  
  WRITE\_STRING\_TWO\_VALUE\_INTG\_-  
    INTG, 890  
  WRITE\_STRING\_TWO\_VALUE\_INTG\_L,  
    890  
  WRITE\_STRING\_TWO\_VALUE\_INTG\_SP,  
    891  
  WRITE\_STRING\_TWO\_VALUE\_INTG\_VS,  
    891  
  WRITE\_STRING\_TWO\_VALUE\_L\_C, 892  
  WRITE\_STRING\_TWO\_VALUE\_L\_DP, 892  
  WRITE\_STRING\_TWO\_VALUE\_L\_INTG,  
    893  
  WRITE\_STRING\_TWO\_VALUE\_L\_L, 893  
  WRITE\_STRING\_TWO\_VALUE\_L\_SP, 894  
  WRITE\_STRING\_TWO\_VALUE\_L\_VS, 894  
  WRITE\_STRING\_TWO\_VALUE\_SP\_C, 895  
  WRITE\_STRING\_TWO\_VALUE\_SP\_DP,  
    895  
  WRITE\_STRING\_TWO\_VALUE\_SP\_INTG,  
    896  
  WRITE\_STRING\_TWO\_VALUE\_SP\_L, 896  
  WRITE\_STRING\_TWO\_VALUE\_SP\_SP,  
    897  
  WRITE\_STRING\_TWO\_VALUE\_SP\_VS,  
    897  
  WRITE\_STRING\_TWO\_VALUE\_VS\_C, 898  
  WRITE\_STRING\_TWO\_VALUE\_VS\_DP,  
    898  
  WRITE\_STRING\_TWO\_VALUE\_VS\_INTG,  
    899  
  WRITE\_STRING\_TWO\_VALUE\_VS\_L, 899  
WRITE\_STRING\_TWO\_VALUE\_VS\_SP,  
  900  
WRITE\_STRING\_TWO\_VALUE\_VS\_VS,  
  900  
INPUT\_OUTPUT::WRITE\_STRING\_VALUE,  
  902  
  WRITE\_STRING\_VALUE\_C, 902  
  WRITE\_STRING\_VALUE\_DP, 902  
  WRITE\_STRING\_VALUE\_INTG, 903  
  WRITE\_STRING\_VALUE\_L, 903  
  WRITE\_STRING\_VALUE\_LINTG, 903  
  WRITE\_STRING\_VALUE\_SP, 904  
  WRITE\_STRING\_VALUE\_VS, 904  
INPUT\_OUTPUT::WRITE\_STRING\_VECTOR,  
  905  
  WRITE\_STRING\_VECTOR\_DP, 905  
  WRITE\_STRING\_VECTOR\_INTG, 905  
  WRITE\_STRING\_VECTOR\_L, 905  
  WRITE\_STRING\_VECTOR\_LINTG, 905  
  WRITE\_STRING\_VECTOR\_SP, 905  
INPUT\_OUTPUT\_MatrixNameIndexFormat  
  WRITE\_STRING\_MATRIX\_NAME\_AND\_-  
    INDICES, 90  
  WRITE\_STRING\_MATRIX\_NAME\_ONLY,  
    90  
INQUIRE\_EOF\_BINARY\_FILE  
  BINARY\_FILE, 215  
INQUIRE\_OPEN\_BINARY\_FILE  
  BINARY\_FILE, 215  
insert\_CH\_CH  
  ISO\_VARYING\_STRING, 501  
  ISO\_VARYING\_STRING::insert, 918  
insert\_CH\_VS  
  ISO\_VARYING\_STRING, 502  
  ISO\_VARYING\_STRING::insert, 918  
INSERT\_TYPE  
  TREES::TREE\_TYPE, 1038  
insert\_VS\_CH  
  ISO\_VARYING\_STRING, 502  
  ISO\_VARYING\_STRING::insert, 918  
insert\_VS\_VS  
  ISO\_VARYING\_STRING, 502  
  ISO\_VARYING\_STRING::insert, 918  
INT\_FORMAT  
  BINARY\_FILE::BINARY\_FILE\_INFO\_-  
    TYPE, 781  
integer  
  f90c\_c.c, 1412  
INTEGER\_SIZE  
  BINARY\_FILE::BINARY\_FILE\_INFO\_-  
    TYPE, 781  
  MACHINE\_CONSTANTS, 541  
INTEGERTYPE  
  binary\_file\_c.c, 1358

INTERNAL\_LIST  
   TYPES::DOMAIN\_MAPPING\_TYPE, 1107

INTERPOLATED\_POINT  
   TYPES::FIELD\_INTERPOLATED\_POINT\_-  
     METRICS\_TYPE, 1206

INTERPOLATION  
   TYPES::EQUATIONS\_TYPE, 1195

INTERPOLATION\_ORDER  
   TYPES::BASIS\_TYPE, 1047

INTERPOLATION\_PARAMETERS  
   TYPES::FIELD\_INTERPOLATED\_POINT\_-  
     TYPE, 1208

INTERPOLATION\_TYPE  
   TYPES::BASIS\_TYPE, 1047  
   TYPES::FIELD\_CREATE\_VALUES\_-  
     CACHE\_TYPE, 1197  
   TYPES::FIELD\_VARIABLE\_-  
     COMPONENT\_TYPE, 1230

INTERPOLATION\_XI  
   TYPES::BASIS\_TYPE, 1047

INTG  
   KINDS\_IntegerKinds, 92

INVERT\_FULL\_DP  
   MATHS, 548  
   MATHS::INVERT, 968

INVERT\_FULL\_SP  
   MATHS, 549  
   MATHS::INVERT, 968

IO1\_FILE\_UNIT  
   BASE\_ROUTINES\_FileUnits, 23

IO2\_FILE\_UNIT  
   BASE\_ROUTINES\_FileUnits, 23

IO3\_FILE\_UNIT  
   BASE\_ROUTINES\_FileUnits, 23

IO4\_FILE\_UNIT  
   BASE\_ROUTINES\_FileUnits, 24

IO5\_FILE\_UNIT  
   BASE\_ROUTINES\_FileUnits, 24

IS\_ABBREVIATION\_C\_C  
   STRINGS, 731  
   STRINGS::IS\_ABBREVIATION, 1019

IS\_ABBREVIATION\_C\_VS  
   STRINGS, 731  
   STRINGS::IS\_ABBREVIATION, 1019

IS\_ABBREVIATION\_VS\_C  
   STRINGS, 732  
   STRINGS::IS\_ABBREVIATION, 1019

IS\_ABBREVIATION\_VS\_VS  
   STRINGS, 732  
   STRINGS::IS\_ABBREVIATION, 1020

IS\_DIGIT  
   STRINGS, 732

IS\_LETTER  
   STRINGS, 732

IS\_LOWERCASE  
   STRINGS, 732

IS\_UPPERCASE  
   STRINGS, 733

IS\_WHITESPACE  
   STRINGS, 733

ISBINARYFILEOPEN  
   BINARY\_FILE::interface, 785

IsBinaryFileOpen  
   binary\_file\_c.c, 1360

ISColoringDestroy  
   CMISS\_PETSC::interface, 801

ISDestroy  
   CMISS\_PETSC::interface, 801

ISENDBINARYFILE  
   BINARY\_FILE::interface, 786

IsEndBinaryFile  
   binary\_file\_c.c, 1360

ISLocalToGlobalMappingApply  
   CMISS\_PETSC::interface, 801

ISLocalToGlobalMappingApplyIS  
   CMISS\_PETSC::interface, 801

ISLocalToGlobalMappingCreate  
   CMISS\_PETSC::interface, 801

ISLocalToGlobalMappingDestroy  
   CMISS\_PETSC::interface, 801

ISLTGMAPPING  
   TYPES::DISTRIBUTED\_MATRIX\_-  
     PETSC\_TYPE, 1071  
   TYPES::DISTRIBUTED\_VECTOR\_-  
     PETSC\_TYPE, 1080

ISO\_VARYING\_STRING, 497  
   adjustl\_, 499  
   adjustr\_, 499  
   char\_auto, 500  
   char\_fixed, 500  
   extract\_CH, 500  
   extract\_VS, 500  
   get\_, 500  
   GET\_BUFFER\_LEN, 511  
   get\_set\_CH, 500  
   get\_set\_VS, 500  
   get\_unit, 500  
   get\_unit\_set\_CH, 501  
   get\_unit\_set\_VS, 501  
   iachar\_, 501  
   ichar\_, 501  
   index\_CH\_VS, 501  
   index\_VS\_CH, 501  
   index\_VS\_VS, 501  
   insert\_CH\_CH, 501  
   insert\_CH\_VS, 502  
   insert\_VS\_CH, 502  
   insert\_VS\_VS, 502

len\_, 502  
len\_trim\_, 502  
lge\_CH\_VS, 502  
lge\_VS\_CH, 502  
lge\_VS\_VS, 502  
lgt\_CH\_VS, 502  
lgt\_VS\_CH, 503  
lgt\_VS\_VS, 503  
lle\_CH\_VS, 503  
lle\_VS\_CH, 503  
lle\_VS\_VS, 503  
llt\_CH\_VS, 503  
llt\_VS\_CH, 503  
llt\_VS\_VS, 503  
op\_assign\_CH\_VS, 503  
op\_assign\_VS\_CH, 504  
op\_concat\_CH\_VS, 504  
op\_concat\_VS\_CH, 504  
op\_concat\_VS\_VS, 504  
op\_eq\_CH\_VS, 504  
op\_eq\_VS\_CH, 504  
op\_eq\_VS\_VS, 504  
op\_ge\_CH\_VS, 505  
op\_ge\_VS\_CH, 505  
op\_ge\_VS\_VS, 505  
op\_gt\_CH\_VS, 505  
op\_gt\_VS\_CH, 505  
op\_gt\_VS\_VS, 505  
op\_le\_CH\_VS, 505  
op\_le\_VS\_CH, 505  
op\_le\_VS\_VS, 505  
op\_lt\_CH\_VS, 506  
op\_lt\_VS\_CH, 506  
op\_lt\_VS\_VS, 506  
op\_ne\_CH\_VS, 506  
op\_ne\_VS\_CH, 506  
op\_ne\_VS\_VS, 506  
put\_CH, 506  
put\_line\_CH, 506  
put\_line\_unit\_CH, 506  
put\_line\_unit\_VS, 507  
put\_line\_VS, 507  
put\_unit\_CH, 507  
put\_unit\_VS, 507  
put\_VS, 507  
remove\_CH, 507  
remove\_VS, 507  
repeat\_, 507  
replace\_CH\_CH\_auto, 507  
replace\_CH\_CH\_CH\_target, 508  
replace\_CH\_CH\_fixed, 508  
replace\_CH\_CH\_VS\_target, 508  
replace\_CH\_VS\_auto, 508  
replace\_CH\_VS\_CH\_target, 508  
replace\_CH\_VS\_fixed, 508  
replace\_CH\_VS\_VS\_target, 508  
replace\_VS\_CH\_auto, 509  
replace\_VS\_CH\_CH\_target, 509  
replace\_VS\_CH\_fixed, 509  
replace\_VS\_CH\_VS\_target, 509  
replace\_VS\_VS\_auto, 509  
replace\_VS\_VS\_CH\_target, 509  
replace\_VS\_VS\_fixed, 509  
replace\_VS\_VS\_VS\_target, 510  
scan\_CH\_VS, 510  
scan\_VS\_CH, 510  
scan\_VS\_VS, 510  
split\_CH, 510  
split\_VS, 510  
trim\_, 510  
var\_str\_, 511  
verify\_CH\_VS, 511  
verify\_VS\_CH, 511  
verify\_VS\_VS, 511  
ISO\_VARYING\_STRING::adjustr, 906  
    adjustr\_, 906  
ISO\_VARYING\_STRING::assignment(=), 907  
    adjustl\_, 907  
    op\_assign\_CH\_VS, 907  
    op\_assign\_VS\_CH, 907  
    op\_concat\_CH\_VS, 907  
    op\_concat\_VS\_CH, 908  
    op\_concat\_VS\_VS, 908  
    op\_eq\_CH\_VS, 908  
    op\_eq\_VS\_CH, 908  
    op\_eq\_VS\_VS, 908  
    op\_ge\_CH\_VS, 908  
    op\_ge\_VS\_CH, 908  
    op\_ge\_VS\_VS, 908  
    op\_gt\_CH\_VS, 908  
    op\_gt\_VS\_CH, 909  
    op\_gt\_VS\_VS, 909  
    op\_le\_CH\_VS, 909  
    op\_le\_VS\_CH, 909  
    op\_le\_VS\_VS, 909  
    op\_lt\_CH\_VS, 909  
    op\_lt\_VS\_CH, 909  
    op\_lt\_VS\_VS, 909  
    op\_ne\_CH\_VS, 909  
    op\_ne\_VS\_CH, 910  
    op\_ne\_VS\_VS, 910  
ISO\_VARYING\_STRING::char, 911  
    char\_auto, 911  
    char\_fixed, 911  
ISO\_VARYING\_STRING::extract, 912  
    extract\_CH, 912  
    extract\_VS, 912  
ISO\_VARYING\_STRING::get, 913

get\_, 913  
 get\_set\_CH, 913  
 get\_set\_VS, 913  
 get\_unit, 913  
 get\_unit\_set\_CH, 913  
 get\_unit\_set\_VS, 913  
 ISO\_VARYING\_STRING::iachar, 915  
 iachar\_, 915  
 ISO\_VARYING\_STRING::ichar, 916  
 ichar\_, 916  
 ISO\_VARYING\_STRING::index, 917  
 index\_CH\_VS, 917  
 index\_VS\_CH, 917  
 index\_VS\_VS, 917  
 ISO\_VARYING\_STRING::insert, 918  
 insert\_CH\_CH, 918  
 insert\_CH\_VS, 918  
 insert\_VS\_CH, 918  
 insert\_VS\_VS, 918  
 ISO\_VARYING\_STRING::len, 919  
 len\_, 919  
 ISO\_VARYING\_STRING::len\_trim, 920  
 len\_trim\_, 920  
 ISO\_VARYING\_STRING::lge, 921  
 lge\_CH\_VS, 921  
 lge\_VS\_CH, 921  
 lge\_VS\_VS, 921  
 ISO\_VARYING\_STRING::lgt, 922  
 lgt\_CH\_VS, 922  
 lgt\_VS\_CH, 922  
 lgt\_VS\_VS, 922  
 ISO\_VARYING\_STRING::lle, 923  
 lle\_CH\_VS, 923  
 lle\_VS\_CH, 923  
 lle\_VS\_VS, 923  
 ISO\_VARYING\_STRING::llt, 924  
 llt\_CH\_VS, 924  
 llt\_VS\_CH, 924  
 llt\_VS\_VS, 924  
 ISO\_VARYING\_STRING::put, 925  
 put\_CH, 925  
 put\_unit\_CH, 925  
 put\_unit\_VS, 925  
 put\_VS, 925  
 ISO\_VARYING\_STRING::put\_line, 926  
 put\_line\_CH, 926  
 put\_line\_unit\_CH, 926  
 put\_line\_unit\_VS, 926  
 put\_line\_VS, 926  
 ISO\_VARYING\_STRING::remove, 927  
 remove\_CH, 927  
 remove\_VS, 927  
 ISO\_VARYING\_STRING::repeat, 928  
 repeat\_, 928  
 ISO\_VARYING\_STRING::replace, 929  
 replace\_CH\_CH\_auto, 929  
 replace\_CH\_CH\_CH\_target, 929  
 replace\_CH\_CH\_fixed, 929  
 replace\_CH\_CH\_VS\_target, 929  
 replace\_CH\_VS\_auto, 930  
 replace\_CH\_VS\_CH\_target, 930  
 replace\_CH\_VS\_fixed, 930  
 replace\_CH\_VS\_VS\_target, 930  
 replace\_VS\_CH\_auto, 930  
 replace\_VS\_CH\_CH\_target, 930  
 replace\_VS\_CH\_fixed, 930  
 replace\_VS\_CH\_VS\_target, 931  
 replace\_VS\_VS\_auto, 931  
 replace\_VS\_VS\_CH\_target, 931  
 replace\_VS\_VS\_fixed, 931  
 replace\_VS\_VS\_VS\_target, 931  
 ISO\_VARYING\_STRING::scan, 932  
 scan\_CH\_VS, 932  
 scan\_VS\_CH, 932  
 scan\_VS\_VS, 932  
 ISO\_VARYING\_STRING::split, 933  
 split\_CH, 933  
 split\_VS, 933  
 ISO\_VARYING\_STRING::trim, 934  
 trim\_, 934  
 ISO\_VARYING\_STRING::var\_str, 935  
 var\_str\_, 935  
 ISO\_VARYING\_STRING::VARYING\_STRING,  
 936  
 chars, 936  
 ISO\_VARYING\_STRING::verify, 937  
 verify\_CH\_VS, 937  
 verify\_VS\_CH, 937  
 verify\_VS\_VS, 937  
 ITERATIVE\_PRECONDITIONER\_TYPE  
 TYPES::LINEAR\_ITERATIVE\_SOLVER\_-  
 TYPE, 1248  
 ITERATIVE\_SOLVER  
 TYPES::LINEAR\_SOLVER\_TYPE, 1250  
 ITERATIVE\_SOLVER\_TYPE  
 TYPES::LINEAR\_ITERATIVE\_SOLVER\_-  
 TYPE, 1248  
 JACOBIAN  
 TYPES::EQUATIONS\_JACOBIAN\_TO\_-  
 VARIABLE\_MAP\_TYPE, 1131  
 TYPES::EQUATIONS\_JACOBIAN\_TYPE,  
 1134  
 TYPES::EQUATIONS\_MATRICES\_-  
 NONLINEAR\_TYPE, 1153  
 TYPES::FIELD\_INTERPOLATED\_POINT\_-  
 METRICS\_TYPE, 1206  
 TYPES::SOLVER\_JACOBIAN\_TYPE, 1320

JACOBIAN\_CALCULATION\_TYPE  
  TYPES::EQUATIONS\_JACOBIAN\_TYPE,  
    1134  
  TYPES::NONLINEAR\_LINESEARCH\_-  
    SOLVER\_TYPE, 1278

JACOBIAN\_COEFFICIENT  
  TYPES::EQUATIONS\_JACOBIAN\_TO\_-  
    VARIABLE\_MAP\_TYPE, 1132

JACOBIAN\_COL\_NUMBER  
  TYPES::SOLVER\_COL\_TO\_EQUATIONS\_-  
    MAP\_TYPE, 1313

JACOBIAN\_COL\_SOLVER\_COLS\_MAP  
  TYPES::JACOBIAN\_TO\_SOLVER\_MAP\_-  
    TYPE, 1244

JACOBIAN\_COUPLING\_COEFFICIENT  
  TYPES::SOLVER\_COL\_TO\_EQUATIONS\_-  
    MAP\_TYPE, 1313

JACOBIAN\_FDCOLORING  
  TYPES::NONLINEAR\_LINESEARCH\_-  
    SOLVER\_TYPE, 1278

JACOBIAN\_ISCOLORING  
  TYPES::NONLINEAR\_LINESEARCH\_-  
    SOLVER\_TYPE, 1278

JACOBIAN\_MATRIX  
  TYPES::JACOBIAN\_TO\_SOLVER\_MAP\_-  
    TYPE, 1244

JACOBIAN\_TO\_SOLVER\_MATRIX\_MAP  
  TYPES::EQUATIONS\_TO\_SOLVER\_-  
    MATRIX\_MAPS\_JM\_TYPE, 1190  
  TYPES::EQUATIONS\_TO\_SOLVER\_-  
    MATRIX\_MAPS\_SM\_TYPE, 1192

JACOBIAN\_TO\_VARIABLE\_MAP  
  TYPES::EQUATIONS\_MAPPING\_-  
    NONLINEAR\_TYPE, 1143

JACOBIAN\_TYPE  
  TYPES::FIELD\_INTERPOLATED\_POINT\_-  
    METRICS\_TYPE, 1206

K0\_DP  
  MATHS, 549  
  MATHS::K0, 969

K0\_SP  
  MATHS, 549  
  MATHS::K0, 969

K1\_DP  
  MATHS, 549  
  MATHS::K1, 970

K1\_SP  
  MATHS, 549  
  MATHS::K1, 970

KDP\_DP  
  MATHS, 549

KDP\_SP  
  MATHS, 549

KEY  
  TREES::TREE\_NODE\_TYPE, 1036

KINDS, 512  
  KINDS::ComplexKinds, 97  
  KINDS::IntegerKinds, 92  
  KINDS::RealKinds, 95  
  KINDS\_ComplexKinds  
    \_DP, 97  
    \_SP, 98  
    DPC, 99  
    SPC, 99

KINDS\_IntegerKinds  
  INTG, 92  
  LINTG, 92  
  PTR, 92  
  SINTG, 94

KINDS\_RealKinds  
  DP, 95  
  QP, 95  
  SP, 95

KSP  
  TYPES::LINEAR\_ITERATIVE\_SOLVER\_-  
    TYPE, 1248

KSPCreate  
  CMISS\_PETSC::interface, 801

KSPDestroy  
  CMISS\_PETSC::interface, 801

KSPGetConvergedReason  
  CMISS\_PETSC::interface, 801

KSPGetIterationNumber  
  CMISS\_PETSC::interface, 801

KSPGetPC  
  CMISS\_PETSC::interface, 802

KSPGetResidualNorm  
  CMISS\_PETSC::interface, 802

KSPSetFromOptions  
  CMISS\_PETSC::interface, 802

KSPSetOperators  
  CMISS\_PETSC::interface, 802

KSPSetTolerances  
  CMISS\_PETSC::interface, 802

KSPSetType  
  CMISS\_PETSC::interface, 802

KSPSetUp  
  CMISS\_PETSC::interface, 802

KSPSolve  
  CMISS\_PETSC::interface, 802

L2NORM\_DP  
  MATHS, 550  
  MATHS::L2NORM, 971

L2NORM\_SP  
  MATHS, 550  
  MATHS::L2NORM, 971

LABEL  
   TYPES::NODE\_TYPE, 1273  
   TYPES::REGION\_TYPE, 1301

LAPACK, 513

LAPACK::interface, 938  
   DGESV, 938

LAPLACE\_EQUATION\_ANALYTIC\_-  
   CALCULATE  
   FIELD\_ROUTINES\_DependentTypes, 62

LAPLACE\_EQUATION\_ANALYTIC\_-  
   PARAMETER\_SET\_UPDATE  
   FIELD\_ROUTINES\_DependentTypes, 62

LAPLACE\_EQUATION\_EQUATIONS\_SET\_-  
   SETUP  
   FIELD\_ROUTINES\_DependentTypes, 63

LAPLACE\_EQUATION\_EQUATIONS\_SET\_-  
   STANDARD\_SETUP  
   FIELD\_ROUTINES\_DependentTypes, 63

LAPLACE\_EQUATION\_EQUATIONS\_SET\_-  
   SUBTYPE\_SET  
   FIELD\_ROUTINES\_DependentTypes, 64

LAPLACE\_EQUATIONFINITE\_ELEMENT\_-  
   CALCULATE  
   FIELD\_ROUTINES\_DependentTypes, 65

LAPLACE\_EQUATION\_FIXED\_-  
   CONDITIONS\_-  
   FIELD\_ROUTINES\_DependentTypes, 65

LAPLACE\_EQUATION\_INTE  
   FIELD\_ROUTINES\_DependentTypes, 66

LAPLACE\_EQUATION\_PROBLEM\_SETUP  
   FIELD\_ROUTINES\_DependentTypes, 67

LAPLACE\_EQUATION\_PROBLEM\_-  
   STANDARD\_SETUP  
   FIELD\_ROUTINES\_DependentTypes, 67

LAPLACE\_EQUATION\_PROBLEM\_-  
   SUBTYPE\_SET  
   FIELD\_ROUTINES\_DependentTypes, 68

LAPLACE\_EQUATION\_THREE\_DIM\_1  
   FIELD\_ROUTINES\_DependentTypes, 69

LAPLACE\_EQUATION\_THREE\_DIM\_2  
   FIELD\_ROUTINES\_DependentTypes, 69

LAPLACE\_EQUATION\_TWO\_DIM\_1  
   FIELD\_ROUTINES\_DependentTypes, 69

LAPLACE\_EQUATION\_TWO\_DIM\_2  
   FIELD\_ROUTINES\_DependentTypes, 70

LAPLACE\_EQUATIONS\_ROUTINES, 514

LEARN\_FILE\_UNIT  
   BASE\_ROUTINES\_FileUnits, 24

LEFT  
   TREES::TREE\_NODE\_TYPE, 1036

len\_  
   ISO\_VARYING\_STRING, 502  
   ISO\_VARYING\_STRING::len, 919

len\_trim\_

ISO\_VARYING\_STRING, 502  
   ISO\_VARYING\_STRING::len\_trim, 920

LENGTHS  
   TYPES::FIELD\_GEOMETRIC\_-  
   PARAMETERS\_TYPE, 1204

lge\_CH\_VS  
   ISO\_VARYING\_STRING, 502  
   ISO\_VARYING\_STRING::lge, 921

lge\_VS\_CH  
   ISO\_VARYING\_STRING, 502  
   ISO\_VARYING\_STRING::lge, 921

lge\_VS\_VS  
   ISO\_VARYING\_STRING, 502  
   ISO\_VARYING\_STRING::lge, 921

lgt\_CH\_VS  
   ISO\_VARYING\_STRING, 502  
   ISO\_VARYING\_STRING::lgt, 922

lgt\_VS\_CH  
   ISO\_VARYING\_STRING, 503  
   ISO\_VARYING\_STRING::lgt, 922

lgt\_VS\_VS  
   ISO\_VARYING\_STRING, 503  
   ISO\_VARYING\_STRING::lgt, 922

LIBRARY\_TYPE  
   TYPES::DISTRIBUTED\_MATRIX\_TYPE,  
   1074

  TYPES::DISTRIBUTED\_VECTOR\_TYPE,  
   1086

  TYPES::SOLVER\_MATRICES\_TYPE, 1323

LINE\_BASES  
   TYPES::BASIS\_TYPE, 1047

LINEAR\_DATA  
   TYPES::EQUATIONS\_TYPE, 1195

LINEAR\_ELASTICITY\_EQUATIONS\_SET\_-  
   SETUP  
   LINEAR\_ELASTICITY\_ROUTINES, 516

LINEAR\_ELASTICITY\_EQUATIONS\_SET\_-  
   SUBTYPE\_SET  
   LINEAR\_ELASTICITY\_ROUTINES, 517

LINEAR\_ELASTICITYFINITE\_ELEMENT\_-  
   CALCULATE  
   LINEAR\_ELASTICITY\_ROUTINES, 518

LINEAR\_ELASTICITY\_PROBLEM\_SETUP  
   LINEAR\_ELASTICITY\_ROUTINES, 518

LINEAR\_ELASTICITY\_PROBLEM\_-  
   SUBTYPE\_SET  
   LINEAR\_ELASTICITY\_ROUTINES, 519

LINEAR\_ELASTICITY\_ROUTINES, 516  
   LINEAR\_ELASTICITY\_EQUATIONS\_-  
   SET\_SETUP, 516

  LINEAR\_ELASTICITY\_EQUATIONS\_-  
   SET\_SUBTYPE\_SET, 517

LINEAR\_ELASTICITYFINITE\_-  
   ELEMENT\_CALCULATE, 518

LINEAR\_ELASTICITY\_PROBLEM\_-  
SETUP, 518  
LINEAR\_ELASTICITY\_PROBLEM\_-  
SUBTYPE\_SET, 519  
LINEAR\_MAPPING  
TYPES::EQUATIONS\_MAPPING\_TYPE,  
1150  
LINEAR\_MATRICES  
TYPES::EQUATIONS\_MATRICES\_TYPE,  
1160  
TYPES::EQUATIONS\_MATRIX\_TYPE,  
1165  
LINEAR\_SOLVE\_TYPE  
TYPES::LINEAR\_SOLVER\_TYPE, 1250  
LINEAR\_SOLVER  
TYPES::LINEAR\_DIRECT\_SOLVER\_-  
TYPE, 1246  
TYPES::LINEAR\_ITERATIVE\_SOLVER\_-  
TYPE, 1248  
TYPES::SOLVER\_TYPE, 1331  
LINEARITY  
TYPES::EQUATIONS\_SET\_TYPE, 1181  
TYPES::SOLUTION\_TYPE, 1310  
LINES  
TYPES::DECOMPOSITION\_LINES\_TYPE,  
1061  
TYPES::DECOMPOSITION\_TOPOLOGY\_-  
TYPE, 1063  
TYPES::DOMAIN\_LINES\_TYPE, 1104  
TYPES::DOMAIN\_TOPOLOGY\_TYPE,  
1118  
LINESEARCH\_ALPHA  
TYPES::NONLINEAR\_LINESEARCH\_-  
SOLVER\_TYPE, 1278  
LINESEARCH\_MAXSTEP  
TYPES::NONLINEAR\_LINESEARCH\_-  
SOLVER\_TYPE, 1278  
LINESEARCH\_SOLVER  
TYPES::NONLINEAR\_SOLVER\_TYPE,  
1280  
LINESEARCH\_STEPTOLERANCE  
TYPES::NONLINEAR\_LINESEARCH\_-  
SOLVER\_TYPE, 1278  
LINESEARCH\_TYPE  
TYPES::NONLINEAR\_LINESEARCH\_-  
SOLVER\_TYPE, 1278  
LINTEGER\_SIZE  
BINARY\_FILE::BINARY\_FILE\_INFO\_-  
TYPE, 781  
LINTG  
KINDS\_IntegerKinds, 92  
LIST\_BUBBLE\_SORT\_METHOD  
LISTSSortingMethod, 104  
LIST\_CREATE\_FINISH  
LISTS, 524  
LIST\_CREATE\_START  
LISTS, 524  
LIST\_DATA\_TYPE\_SET  
LISTS, 525  
LIST\_DESTROY  
LISTS, 525  
LIST\_DETACH\_AND\_DESTROY\_DP  
LISTS, 526  
LISTS::LIST\_DETACH\_AND\_DESTROY,  
939  
LIST\_DETACH\_AND\_DESTROY\_INTG  
LISTS, 526  
LISTS::LIST\_DETACH\_AND\_DESTROY,  
939  
LIST\_DETACH\_AND\_DESTROY\_SP  
LISTS, 527  
LISTS::LIST\_DETACH\_AND\_DESTROY,  
940  
LIST\_DP  
LISTS::LIST\_TYPE, 958  
LIST\_DP\_TYPE  
LISTS\_DataType, 100  
LIST\_FINALISE  
LISTS, 527  
LIST\_FINISHED  
LISTS::LIST\_TYPE, 958  
LIST\_HEAP\_SORT\_METHOD  
LISTSSortingMethod, 104  
LIST\_INITIAL\_SIZE\_SET  
LISTS, 528  
LIST\_INITIALISE  
LISTS, 528  
LIST\_INTG  
LISTS::LIST\_TYPE, 958  
LIST\_INTG\_TYPE  
LISTS\_DataType, 100  
LIST\_ITEM\_ADD\_DP1  
LISTS, 528  
LISTS::LIST\_ITEM\_ADD, 941  
LIST\_ITEM\_ADD\_INTG1  
LISTS, 529  
LISTS::LIST\_ITEM\_ADD, 941  
LIST\_ITEM\_ADD\_SP1  
LISTS, 529  
LISTS::LIST\_ITEM\_ADD, 941  
LIST\_ITEM\_DELETE  
LISTS, 530  
LIST\_ITEM\_IN\_LIST\_DP1  
LISTS, 530  
LISTS::LIST\_ITEM\_IN\_LIST, 943  
LIST\_ITEM\_IN\_LIST\_INTG1  
LISTS, 530  
LISTS::LIST\_ITEM\_IN\_LIST, 943

LIST\_ITEM\_IN\_LIST\_SP1  
     LISTS, 531  
     LISTS::LIST\_ITEM\_IN\_LIST, 944  
 LIST\_NUMBER\_OF\_ITEMS\_GET  
     LISTS, 531  
 LIST\_OF\_GLOBAL\_NUMBER  
     FIELD\_IO\_ROUTINES::FIELD\_IO\_-  
         ELEMENTALL\_INFO\_SET, 845  
     FIELD\_IO\_ROUTINES::FIELD\_IO\_-  
         NODAL\_INFO\_SET, 848  
 LIST\_REMOVE\_DUPLICATES  
     LISTS, 532  
 LIST\_SEARCH\_DP\_ARRAY  
     LISTS, 532  
     LISTS::LIST\_SEARCH, 946  
 LIST\_SEARCH\_INTG\_ARRAY  
     LISTS, 533  
     LISTS::LIST\_SEARCH, 946  
 LIST\_SEARCH\_LINEAR\_DP\_ARRAY  
     LISTS, 533  
     LISTS::LIST\_SEARCH\_LINEAR, 948  
 LIST\_SEARCH\_LINEAR\_INTG\_ARRAY  
     LISTS, 534  
     LISTS::LIST\_SEARCH\_LINEAR, 948  
 LIST\_SEARCH\_LINEAR\_SP\_ARRAY  
     LISTS, 534  
     LISTS::LIST\_SEARCH\_LINEAR, 948  
 LIST\_SEARCH\_SP\_ARRAY  
     LISTS, 535  
     LISTS::LIST\_SEARCH, 947  
 LIST\_SHELL\_SORT\_METHOD  
     LISTSSortingMethod, 104  
 LIST\_SORT\_ASCENDING\_TYPE  
     LISTS\_SortingOrder, 102  
 LIST\_SORT\_BUBBLE\_DP\_ARRAY  
     LISTS, 535  
     LISTS::LIST\_SORT\_BUBBLE, 952  
 LIST\_SORT\_BUBBLE\_INTG\_ARRAY  
     LISTS, 535  
     LISTS::LIST\_SORT\_BUBBLE, 952  
 LIST\_SORT\_BUBBLE\_SP\_ARRAY  
     LISTS, 536  
     LISTS::LIST\_SORT\_BUBBLE, 952  
 LIST\_SORT\_DESCENDING\_TYPE  
     LISTS\_SortingOrder, 102  
 LIST\_SORT\_DP\_ARRAY  
     LISTS, 536  
     LISTS::LIST\_SORT, 950  
 LIST\_SORT\_HEAP\_DP\_ARRAY  
     LISTS, 536  
     LISTS::LIST\_SORT\_HEAP, 954  
 LIST\_SORT\_HEAP\_INTG\_ARRAY  
     LISTS, 537  
     LISTS::LIST\_SORT\_HEAP, 954  
 LIST\_SORT\_HEAP\_SP\_ARRAY  
     LISTS, 538  
     LISTS::LIST\_SORT\_HEAP, 954  
 LIST\_SORT\_INTG\_ARRAY  
     LISTS, 537  
     LISTS::LIST\_SORT, 950  
 LIST\_SORT\_SHELL\_DP\_ARRAY  
     LISTS, 538  
     LISTS::LIST\_SORT\_SHELL, 956  
 LIST\_SORT\_SHELL\_INTG\_ARRAY  
     LISTS, 538  
     LISTS::LIST\_SORT\_SHELL, 956  
 LIST\_SORT\_SHELL\_SP\_ARRAY  
     LISTS, 538  
     LISTS::LIST\_SORT\_SHELL, 956  
 LIST\_SORT\_SP\_ARRAY  
     LISTS, 539  
     LISTS::LIST\_SORT, 950  
 LIST\_SP  
     LISTS::LIST\_TYPE, 958  
 LIST\_SP\_TYPE  
     LISTS\_DataType, 101  
 LIST\_TO\_CHARACTER\_C  
     STRINGS, 733  
     STRINGS::LIST\_TO\_CHARACTER, 1021  
 LIST\_TO\_CHARACTER\_DP  
     STRINGS, 734  
     STRINGS::LIST\_TO\_CHARACTER, 1021  
 LIST\_TO\_CHARACTER\_INTG  
     STRINGS, 734  
     STRINGS::LIST\_TO\_CHARACTER, 1022  
 LIST\_TO\_CHARACTER\_L  
     STRINGS, 735  
     STRINGS::LIST\_TO\_CHARACTER, 1022  
 LIST\_TO\_CHARACTER\_LINTG  
     STRINGS, 735  
     STRINGS::LIST\_TO\_CHARACTER, 1022  
 LIST\_TO\_CHARACTER\_SP  
     STRINGS, 736  
     STRINGS::LIST\_TO\_CHARACTER, 1023  
 LIST\_UNSORTED\_TYPE  
     LISTS\_SortingOrder, 102  
 LISTS, 520  
     LIST\_CREATE\_FINISH, 524  
     LIST\_CREATE\_START, 524  
     LIST\_DATA\_TYPE\_SET, 525  
     LIST\_DESTROY, 525  
     LIST\_DETACH\_AND\_DESTROY\_DP, 526  
     LIST\_DETACH\_AND\_DESTROY\_INTG,  
         526  
     LIST\_DETACH\_AND\_DESTROY\_SP, 527  
     LIST\_FINALISE, 527  
     LIST\_INITIAL\_SIZE\_SET, 528  
     LIST\_INITIALISE, 528

LIST\_ITEM\_ADD\_DP1, 528  
LIST\_ITEM\_ADD\_INTG1, 529  
LIST\_ITEM\_ADD\_SP1, 529  
LIST\_ITEM\_DELETE, 530  
LIST\_ITEM\_IN\_LIST\_DP1, 530  
LIST\_ITEM\_IN\_LIST\_INTG1, 530  
LIST\_ITEM\_IN\_LIST\_SP1, 531  
LIST\_NUMBER\_OF\_ITEMS\_GET, 531  
LIST\_REMOVE\_DUPLICATES, 532  
LIST\_SEARCH\_DP\_ARRAY, 532  
LIST\_SEARCH\_INTG\_ARRAY, 533  
LIST\_SEARCH\_LINEAR\_DP\_ARRAY, 533  
LIST\_SEARCH\_LINEAR\_INTG\_ARRAY,  
  534  
LIST\_SEARCH\_LINEAR\_SP\_ARRAY, 534  
LIST\_SEARCH\_SP\_ARRAY, 535  
LIST\_SORT\_BUBBLE\_DP\_ARRAY, 535  
LIST\_SORT\_BUBBLE\_INTG\_ARRAY, 535  
LIST\_SORT\_BUBBLE\_SP\_ARRAY, 536  
LIST\_SORT\_DP\_ARRAY, 536  
LIST\_SORT\_HEAP\_DP\_ARRAY, 536  
LIST\_SORT\_HEAP\_INTG\_ARRAY, 537  
LIST\_SORT\_HEAP\_SP\_ARRAY, 537  
LIST\_SORT\_INTG\_ARRAY, 537  
LIST\_SORT\_SHELL\_DP\_ARRAY, 538  
LIST\_SORT\_SHELL\_INTG\_ARRAY, 538  
LIST\_SORT\_SHELL\_SP\_ARRAY, 538  
LIST\_SORT\_SP\_ARRAY, 539  
LISTS::DataType, 100  
LISTS::LIST\_DETACH\_AND\_DESTROY, 939  
  LIST\_DETACH\_AND\_DESTROY\_DP, 939  
  LIST\_DETACH\_AND\_DESTROY\_INTG,  
    939  
  LIST\_DETACH\_AND\_DESTROY\_SP, 940  
LISTS::LIST\_ITEM\_ADD, 941  
  LIST\_ITEM\_ADD\_DP1, 941  
  LIST\_ITEM\_ADD\_INTG1, 941  
  LIST\_ITEM\_ADD\_SP1, 941  
LISTS::LIST\_ITEM\_IN\_LIST, 943  
  LIST\_ITEM\_IN\_LIST\_DP1, 943  
  LIST\_ITEM\_IN\_LIST\_INTG1, 943  
  LIST\_ITEM\_IN\_LIST\_SP1, 944  
LISTS::LIST\_PTR\_TYPE, 945  
  PTR, 945  
LISTS::LIST\_SEARCH, 946  
  LIST\_SEARCH\_DP\_ARRAY, 946  
  LIST\_SEARCH\_INTG\_ARRAY, 946  
  LIST\_SEARCH\_SP\_ARRAY, 947  
LISTS::LIST\_SEARCH\_LINEAR, 948  
  LIST\_SEARCH\_LINEAR\_DP\_ARRAY, 948  
  LIST\_SEARCH\_LINEAR\_INTG\_ARRAY,  
    948  
  LIST\_SEARCH\_LINEAR\_SP\_ARRAY, 948  
LISTS::LIST\_SORT, 950  
  LIST\_SORT\_DP\_ARRAY, 950  
  LIST\_SORT\_INTG\_ARRAY, 950  
  LIST\_SORT\_SP\_ARRAY, 950  
LISTS::LIST\_SORT\_BUBBLE, 952  
  LIST\_SORT\_BUBBLE\_DP\_ARRAY, 952  
  LIST\_SORT\_BUBBLE\_INTG\_ARRAY, 952  
  LIST\_SORT\_BUBBLE\_SP\_ARRAY, 952  
LISTS::LIST\_SORT\_HEAP, 954  
  LIST\_SORT\_HEAP\_DP\_ARRAY, 954  
  LIST\_SORT\_HEAP\_INTG\_ARRAY, 954  
  LIST\_SORT\_HEAP\_SP\_ARRAY, 954  
LISTS::LIST\_SORT\_SHELL, 956  
  LIST\_SORT\_SHELL\_DP\_ARRAY, 956  
  LIST\_SORT\_SHELL\_INTG\_ARRAY, 956  
  LIST\_SORT\_SHELL\_SP\_ARRAY, 956  
LISTS::LIST\_TYPE, 957  
  DATA\_TYPE, 957  
  INITIAL\_SIZE, 958  
  LIST\_DP, 958  
  LIST\_FINISHED, 958  
  LIST\_INTG, 958  
  LIST\_SP, 958  
  NUMBER\_IN\_LIST, 958  
  SIZE, 958  
  SORT\_METHOD, 958  
  SORT\_ORDER, 959  
LISTS::SortingMethod, 104  
LISTS::SortingOrder, 102  
LISTS\_DataType  
  LIST\_DP\_TYPE, 100  
  LIST\_INTG\_TYPE, 100  
  LIST\_SP\_TYPE, 101  
LISTS\_SortingOrder  
  LIST\_SORT\_ASCENDING\_TYPE, 102  
  LIST\_SORT\_DESCENDING\_TYPE, 102  
  LIST\_UNSORTED\_TYPE, 102  
LISTSSortingMethod  
  LIST\_BUBBLE\_SORT\_METHOD, 104  
  LIST\_HEAP\_SORT\_METHOD, 104  
  LIST\_SHELL\_SORT\_METHOD, 104  
lle\_CH\_VS  
  ISO\_VARYING\_STRING, 503  
  ISO\_VARYING\_STRING::lle, 923  
lle\_VS\_CH  
  ISO\_VARYING\_STRING, 503  
  ISO\_VARYING\_STRING::lle, 923  
lle\_VS\_VS  
  ISO\_VARYING\_STRING, 503  
  ISO\_VARYING\_STRING::lle, 923  
llt\_CH\_VS  
  ISO\_VARYING\_STRING, 503  
  ISO\_VARYING\_STRING::llt, 924  
llt\_VS\_CH  
  ISO\_VARYING\_STRING, 503

ISO\_VARYING\_STRING::llt, 924  
 llt\_VS\_VS  
   ISO\_VARYING\_STRING, 503  
   ISO\_VARYING\_STRING::llt, 924  
 LOCAL\_GHOST\_RECEIVE\_INDICES  
   TYPES::DOMAIN\_ADJACENT\_-  
     DOMAIN\_TYPE, 1087  
 LOCAL\_GHOST\_SEND\_INDICES  
   TYPES::DOMAIN\_ADJACENT\_-  
     DOMAIN\_TYPE, 1087  
 LOCAL\_LINE\_XI\_DIRECTION  
   TYPES::BASIS\_TYPE, 1048  
 LOCAL\_NUMBER  
   TYPES::DECOMPOSITION\_ELEMENT\_-  
     TYPE, 1056  
   TYPES::DOMAIN\_GLOBAL\_MAPPING\_-  
     TYPE, 1099  
   TYPES::DOMAIN\_NODE\_TYPE, 1112  
 LOCAL\_TO\_GLOBAL\_MAP  
   TYPES::DOMAIN\_MAPPING\_TYPE, 1107  
 LOCAL\_TYPE  
   TYPES::DOMAIN\_GLOBAL\_MAPPING\_-  
     TYPE, 1099  
 logical  
   binary\_file\_c.c, 1359  
   f90c\_c.c, 1412  
 LOGICAL\_SIZE  
   BINARY\_FILE::BINARY\_FILE\_INFO\_-  
     TYPE, 781  
   MACHINE\_CONSTANTS, 541  
 LOGICAL\_TO\_CHARACTER  
   STRINGS, 736  
 LOGICAL\_TO\_VSTRING  
   STRINGS, 736  
 LOGICALTYPE  
   binary\_file\_c.c, 1358  
 LONG\_INTEGER\_SIZE  
   MACHINE\_CONSTANTS, 541  
 LONGINTTYPE  
   binary\_file\_c.c, 1358

M

TYPES::DISTRIBUTED\_MATRIX\_-  
   PETSC\_TYPE, 1071  
 TYPES::MATRIX\_TYPE, 1254  
 MACHINE\_CHAR\_FORMAT  
   MACHINE\_CONSTANTS, 541  
 MACHINE\_CONSTANTS, 540  
   CHARACTER\_SIZE, 540  
   DOUBLE\_COMPLEX\_SIZE, 540  
   DOUBLE\_REAL\_SIZE, 540  
   ERROR\_SEPARATOR\_CONSTANT, 541  
   INTEGER\_SIZE, 541  
   LOGICAL\_SIZE, 541

LONG\_INTEGER\_SIZE, 541  
 MACHINE\_CHAR\_FORMAT, 541  
 MACHINE\_DP\_FORMAT, 541  
 MACHINE\_ENDIAN, 541  
 MACHINE\_INT\_FORMAT, 542  
 MACHINE\_OS, 542  
 MACHINE\_SP\_FORMAT, 542  
 MACHINE\_TYPE, 542  
 SHORT\_INTEGER\_SIZE, 542  
 SINGLE\_COMPLEX\_SIZE, 542  
 SINGLE\_REAL\_SIZE, 542  
 MACHINE\_DP\_FORMAT  
   MACHINE\_CONSTANTS, 541  
 MACHINE\_ENDIAN  
   MACHINE\_CONSTANTS, 541  
 MACHINE\_INT\_FORMAT  
   MACHINE\_CONSTANTS, 542  
 MACHINE\_OS  
   MACHINE\_CONSTANTS, 542  
 MACHINE\_SP\_FORMAT  
   MACHINE\_CONSTANTS, 542  
 MACHINE\_TYPE  
   BINARY\_FILE::BINARY\_FILE\_INFO\_-  
     TYPE, 781  
   MACHINE\_CONSTANTS, 542

MAPPINGS

TYPES::DOMAIN\_TYPE, 1119  
 TYPES::FIELD\_TYPE, 1227  
 MatAssemblyBegin  
   CMISS\_PETSC::interface, 802  
 MatAssemblyEnd  
   CMISS\_PETSC::interface, 802  
 MatCreate  
   CMISS\_PETSC::interface, 803  
 MatCreateMPIAIJ  
   CMISS\_PETSC::interface, 803  
 MatCreateMPIDense  
   CMISS\_PETSC::interface, 803  
 MatCreateSeqAIJ  
   CMISS\_PETSC::interface, 803  
 MatCreateSeqDense  
   CMISS\_PETSC::interface, 803  
 MatDestroy  
   CMISS\_PETSC::interface, 803

MATERIAL\_FIELD

TYPES::EQUATIONS\_INTERPOLATION\_-  
   TYPE, 1129  
 TYPES::EQUATIONS\_SET\_MATERIALS\_-  
   TYPE, 1174  
 MATERIAL\_INTERP\_PARAMETERS  
   TYPES::EQUATIONS\_INTERPOLATION\_-  
     TYPE, 1129  
 MATERIAL\_INTERP\_POINT

TYPES::EQUATIONS\_INTERPOLATION\_-  
TYPE, 1129

MATERIALS  
  TYPES::EQUATIONS\_SET\_TYPE, 1181

MATERIALS\_FINISHED  
  TYPES::EQUATIONS\_SET\_MATERIALS\_-  
  TYPE, 1174

MatFDColoringCreate  
  CMISS\_PETSC::interface, 803

MatFDColoringDestroy  
  CMISS\_PETSC::interface, 803

MatFDColoringSetFromOptions  
  CMISS\_PETSC::interface, 803

MatGetArray  
  CMISS\_PETSC::interface, 803

MatGetArrayF90  
  CMISS\_PETSC::interface, 804

MatGetColoring  
  CMISS\_PETSC::interface, 804

MatGetOwnershipRange  
  CMISS\_PETSC::interface, 804

MatGetValues  
  CMISS\_PETSC::interface, 804

MATHS, 544  
  CROSS\_PRODUCT\_DP, 545  
  CROSS\_PRODUCT\_INTG, 545  
  CROSS\_PRODUCT\_SP, 545  
  D\_CROSS\_PRODUCT\_DP, 546  
  D\_CROSS\_PRODUCT\_INTG, 546  
  D\_CROSS\_PRODUCT\_SP, 546  
  DETERMINANT\_FULL\_DP, 546  
  DETERMINANT\_FULL\_INTG, 547  
  DETERMINANT\_FULL\_SP, 547  
  EDP\_DP, 547  
  EDP\_SP, 547  
  EIGENVALUE\_FULL\_DP, 547  
  EIGENVALUE\_FULL\_SP, 547  
  EIGENVECTOR\_FULL\_DP, 548  
  EIGENVECTOR\_FULL\_SP, 548  
  I0\_DP, 548  
  I0\_SP, 548  
  I1\_DP, 548  
  I1\_SP, 548  
  INVERT\_FULL\_DP, 548  
  INVERT\_FULL\_SP, 549  
  K0\_DP, 549  
  K0\_SP, 549  
  K1\_DP, 549  
  K1\_SP, 549  
  KDP\_DP, 549  
  KDP\_SP, 549  
  L2NORM\_DP, 550  
  L2NORM\_SP, 550  
  MATRIX\_PRODUCT\_DP, 550  
  MATRIX\_PRODUCT\_SP, 550  
  NORMALISE\_DP, 551  
  NORMALISE\_SP, 551  
  SOLVE\_SMALL\_LINEAR\_SYSTEM\_DP,  
    551  
  SOLVE\_SMALL\_LINEAR\_SYSTEM\_SP,  
    551

MATHS::CROSS\_PRODUCT, 960  
  CROSS\_PRODUCT\_DP, 960  
  CROSS\_PRODUCT\_INTG, 960  
  CROSS\_PRODUCT\_SP, 960

MATHS::D\_CROSS\_PRODUCT, 961  
  D\_CROSS\_PRODUCT\_DP, 961  
  D\_CROSS\_PRODUCT\_INTG, 961  
  D\_CROSS\_PRODUCT\_SP, 961

MATHS::DETERMINANT, 962  
  DETERMINANT\_FULL\_DP, 962  
  DETERMINANT\_FULL\_INTG, 962  
  DETERMINANT\_FULL\_SP, 962

MATHS::EDP, 963  
  EDP\_DP, 963  
  EDP\_SP, 963

MATHS::EIGENVALUE, 964  
  EIGENVALUE\_FULL\_DP, 964  
  EIGENVALUE\_FULL\_SP, 964

MATHS::EIGENVECTOR, 965  
  EIGENVECTOR\_FULL\_DP, 965  
  EIGENVECTOR\_FULL\_SP, 965

MATHS::I0, 966  
  I0\_DP, 966  
  I0\_SP, 966

MATHS::I1, 967  
  I1\_DP, 967  
  I1\_SP, 967

MATHS::INVERT, 968  
  INVERT\_FULL\_DP, 968  
  INVERT\_FULL\_SP, 968

MATHS::K0, 969  
  K0\_DP, 969  
  K0\_SP, 969

MATHS::K1, 970  
  K1\_DP, 970  
  K1\_SP, 970

MATHS::L2NORM, 971  
  L2NORM\_DP, 971  
  L2NORM\_SP, 971

MATHS::MATRIX\_PRODUCT, 972  
  MATRIX\_PRODUCT\_DP, 972  
  MATRIX\_PRODUCT\_SP, 972

MATHS::MATRIX\_TRANSPOSE, 973  
  MATRIX\_TRANSPOSE\_DP, 973  
  MATRIX\_TRANSPOSE\_SP, 973

MATHS::NORMALISE, 974  
 NORMALISE\_DP, 974  
 NORMALISE\_SP, 974

MATHS::SOLVE\_SMALL\_LINEAR\_SYSTEM, 975  
 SOLVE\_SMALL\_LINEAR\_SYSTEM\_DP, 975  
 SOLVE\_SMALL\_LINEAR\_SYSTEM\_SP, 975

MatRestoreArray  
 CMISS\_PETSC::interface, 804

MatRestoreArrayF90  
 CMISS\_PETSC::interface, 804

MATRICES  
 TYPES::EQUATIONS\_MATRICES\_LINEAR\_TYPE, 1152  
 TYPES::SOLVER\_MATRICES\_TYPE, 1323

MATRIX  
 TYPES::DISTRIBUTED\_MATRIX\_CMISS\_TYPE, 1068  
 TYPES::DISTRIBUTED\_MATRIX\_PETSC\_TYPE, 1071  
 TYPES::ELEMENT\_MATRIX\_TYPE, 1122  
 TYPES::EQUATIONS\_MATRIX\_TYPE, 1166  
 TYPES::SOLVER\_MATRIX\_TYPE, 1326

MATRIX\_ALL\_VALUES\_SET\_DP  
 MATRIX\_VECTOR, 560  
 MATRIX\_VECTOR::MATRIX\_ALL\_VALUES\_SET, 976

MATRIX\_ALL\_VALUES\_SET\_INTG  
 MATRIX\_VECTOR, 560  
 MATRIX\_VECTOR::MATRIX\_ALL\_VALUES\_SET, 976

MATRIX\_ALL\_VALUES\_SET\_L  
 MATRIX\_VECTOR, 561  
 MATRIX\_VECTOR::MATRIX\_ALL\_VALUES\_SET, 976

MATRIX\_ALL\_VALUES\_SET\_SP  
 MATRIX\_VECTOR, 561  
 MATRIX\_VECTOR::MATRIX\_ALL\_VALUES\_SET, 977

MATRIX\_BLOCK\_STORAGE\_TYPE  
 MATRIX\_VECTOR\_StorageTypes, 109

MATRIX\_COEFFICIENT  
 TYPES::EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_TYPE, 1164

MATRIX\_COEFFICIENTS  
 TYPES::EQUATIONS\_MAPPING\_CREATE\_VALUES\_CACHE\_TYPE, 1137

MATRIX\_COLUMN\_MAJOR\_STORAGE\_TYPE  
 MATRIX\_VECTOR\_StorageTypes, 110

MATRIX\_COMPRESSED\_COLUMN\_STORAGE\_TYPE  
 MATRIX\_VECTOR\_StorageTypes, 110

MATRIX\_COMPRESSED\_ROW\_STORAGE\_TYPE  
 MATRIX\_VECTOR\_StorageTypes, 110

MATRIX\_CREATE\_FINISH  
 MATRIX\_VECTOR, 561

MATRIX\_CREATE\_START  
 MATRIX\_VECTOR, 562

MATRIX\_DATA\_GET\_DP  
 MATRIX\_VECTOR, 562  
 MATRIX\_VECTOR::MATRIX\_DATA\_GET, 978

MATRIX\_DATA\_GET\_INTG  
 MATRIX\_VECTOR, 563  
 MATRIX\_VECTOR::MATRIX\_DATA\_GET, 978

MATRIX\_DATA\_GET\_L  
 MATRIX\_VECTOR, 563  
 MATRIX\_VECTOR::MATRIX\_DATA\_GET, 978

MATRIX\_DATA\_GET\_SP  
 MATRIX\_VECTOR, 563  
 MATRIX\_VECTOR::MATRIX\_DATA\_GET, 979

MATRIX\_DATA\_TYPE\_SET  
 MATRIX\_VECTOR, 564

MATRIX\_DESTROY  
 MATRIX\_VECTOR, 564

MATRIX\_DIAGONAL\_STORAGE\_TYPE  
 MATRIX\_VECTOR\_StorageTypes, 111

MATRIX\_DUPLICATE  
 MATRIX\_VECTOR, 565

MATRIX\_FINALISE  
 MATRIX\_VECTOR, 565

MATRIX\_FINISHED  
 TYPES::DISTRIBUTED\_MATRIX\_TYPE, 1074  
 TYPES::MATRIX\_TYPE, 1254

MATRIX\_INITIALISE  
 MATRIX\_VECTOR, 566

MATRIX\_MAX\_COLUMNS\_PER\_ROW\_GET  
 MATRIX\_VECTOR, 566

MATRIX\_MAX\_SIZE\_SET  
 MATRIX\_VECTOR, 566

MATRIX\_NUMBER  
 TYPES::EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_TYPE, 1164  
 TYPES::EQUATIONS\_MATRIX\_TYPE, 1166  
 TYPES::SOLVER\_MATRIX\_TYPE, 1326

MATRIX\_NUMBER\_NON\_ZEROS\_SET  
 MATRIX\_VECTOR, 567

MATRIX\_OUTPUT  
    MATRIX\_VECTOR, 567

MATRIX\_PRODUCT\_DP  
    MATHS, 550  
    MATHS::MATRIX\_PRODUCT, 972

MATRIX\_PRODUCT\_SP  
    MATHS, 550  
    MATHS::MATRIX\_PRODUCT, 972

MATRIX\_ROW\_COLUMN\_STORAGE\_TYPE  
    MATRIX\_VECTOR\_StorageTypes, 111

MATRIX\_ROW\_MAJOR\_STORAGE\_TYPE  
    MATRIX\_VECTOR\_StorageTypes, 111

MATRIX\_SIZE\_SET  
    MATRIX\_VECTOR, 568

MATRIX\_STORAGE\_LOCATION\_FIND  
    MATRIX\_VECTOR, 568

MATRIX\_STORAGE\_LOCATIONS\_GET  
    MATRIX\_VECTOR, 569

MATRIX\_STORAGE\_LOCATIONS\_SET  
    MATRIX\_VECTOR, 570

MATRIX\_STORAGE\_TYPE\_GET  
    MATRIX\_VECTOR, 570

MATRIX\_STORAGE\_TYPE\_SET  
    MATRIX\_VECTOR, 571

MATRIX\_TRANSPOSE\_DP  
    MATHS, 550  
    MATHS::MATRIX\_TRANSPOSE, 973

MATRIX\_TRANSPOSE\_SP  
    MATHS, 550  
    MATHS::MATRIX\_TRANSPOSE, 973

MATRIX\_VALUES\_ADD\_DP  
    MATRIX\_VECTOR, 571  
    MATRIX\_VECTOR::MATRIX\_VALUES\_-  
        ADD, 980

MATRIX\_VALUES\_ADD\_DP1  
    MATRIX\_VECTOR, 572  
    MATRIX\_VECTOR::MATRIX\_VALUES\_-  
        ADD, 981

MATRIX\_VALUES\_ADD\_DP2  
    MATRIX\_VECTOR, 572  
    MATRIX\_VECTOR::MATRIX\_VALUES\_-  
        ADD, 981

MATRIX\_VALUES\_ADD\_INTG  
    MATRIX\_VECTOR, 573  
    MATRIX\_VECTOR::MATRIX\_VALUES\_-  
        ADD, 981

MATRIX\_VALUES\_ADD\_INTG1  
    MATRIX\_VECTOR, 573  
    MATRIX\_VECTOR::MATRIX\_VALUES\_-  
        ADD, 982

MATRIX\_VALUES\_ADD\_INTG2  
    MATRIX\_VECTOR, 574  
    MATRIX\_VECTOR::MATRIX\_VALUES\_-  
        ADD, 982

MATRIX\_VALUES\_ADD\_L  
    MATRIX\_VECTOR, 574

MATRIX\_VECTOR::MATRIX\_VALUES\_-  
    ADD, 982

MATRIX\_VALUES\_ADD\_L1  
    MATRIX\_VECTOR, 575

MATRIX\_VECTOR::MATRIX\_VALUES\_-  
    ADD, 983

MATRIX\_VALUES\_ADD\_L2  
    MATRIX\_VECTOR, 575

MATRIX\_VECTOR::MATRIX\_VALUES\_-  
    ADD, 983

MATRIX\_VALUES\_ADD\_SP  
    MATRIX\_VECTOR, 576  
    MATRIX\_VECTOR::MATRIX\_VALUES\_-  
        ADD, 983

MATRIX\_VALUES\_ADD\_SP1  
    MATRIX\_VECTOR, 576  
    MATRIX\_VECTOR::MATRIX\_VALUES\_-  
        ADD, 984

MATRIX\_VALUES\_ADD\_SP2  
    MATRIX\_VECTOR, 577  
    MATRIX\_VECTOR::MATRIX\_VALUES\_-  
        ADD, 984

MATRIX\_VALUES\_GET\_DP  
    MATRIX\_VECTOR, 577  
    MATRIX\_VECTOR::MATRIX\_VALUES\_-  
        GET, 985

MATRIX\_VALUES\_GET\_DP1  
    MATRIX\_VECTOR, 578  
    MATRIX\_VECTOR::MATRIX\_VALUES\_-  
        GET, 986

MATRIX\_VALUES\_GET\_DP2  
    MATRIX\_VECTOR, 578  
    MATRIX\_VECTOR::MATRIX\_VALUES\_-  
        GET, 986

MATRIX\_VALUES\_GET\_INTG  
    MATRIX\_VECTOR, 579  
    MATRIX\_VECTOR::MATRIX\_VALUES\_-  
        GET, 986

MATRIX\_VALUES\_GET\_INTG1  
    MATRIX\_VECTOR, 579  
    MATRIX\_VECTOR::MATRIX\_VALUES\_-  
        GET, 987

MATRIX\_VALUES\_GET\_INTG2  
    MATRIX\_VECTOR, 580  
    MATRIX\_VECTOR::MATRIX\_VALUES\_-  
        GET, 987

MATRIX\_VALUES\_GET\_L  
    MATRIX\_VECTOR, 580  
    MATRIX\_VECTOR::MATRIX\_VALUES\_-  
        GET, 987

MATRIX\_VALUES\_GET\_L1  
    MATRIX\_VECTOR, 581

MATRIX\_VECTOR::MATRIX\_VALUES\_-  
GET, 988

MATRIX\_VALUES\_GET\_L2  
  MATRIX\_VECTOR, 581

MATRIX\_VECTOR::MATRIX\_VALUES\_-  
GET, 988

MATRIX\_VALUES\_GET\_SP  
  MATRIX\_VECTOR, 582

MATRIX\_VECTOR::MATRIX\_VALUES\_-  
GET, 988

MATRIX\_VALUES\_GET\_SP1  
  MATRIX\_VECTOR, 582

MATRIX\_VECTOR::MATRIX\_VALUES\_-  
GET, 989

MATRIX\_VALUES\_GET\_SP2  
  MATRIX\_VECTOR, 583

MATRIX\_VECTOR::MATRIX\_VALUES\_-  
GET, 989

MATRIX\_VALUES\_SET\_DP  
  MATRIX\_VECTOR, 583

MATRIX\_VECTOR::MATRIX\_VALUES\_-  
SET, 990

MATRIX\_VALUES\_SET\_DP1  
  MATRIX\_VECTOR, 584

MATRIX\_VECTOR::MATRIX\_VALUES\_-  
SET, 991

MATRIX\_VALUES\_SET\_DP2  
  MATRIX\_VECTOR, 584

MATRIX\_VECTOR::MATRIX\_VALUES\_-  
SET, 991

MATRIX\_VALUES\_SET\_INTG  
  MATRIX\_VECTOR, 585

MATRIX\_VECTOR::MATRIX\_VALUES\_-  
SET, 991

MATRIX\_VALUES\_SET\_INTG1  
  MATRIX\_VECTOR, 585

MATRIX\_VECTOR::MATRIX\_VALUES\_-  
SET, 992

MATRIX\_VALUES\_SET\_INTG2  
  MATRIX\_VECTOR, 586

MATRIX\_VECTOR::MATRIX\_VALUES\_-  
SET, 992

MATRIX\_VALUES\_SET\_L  
  MATRIX\_VECTOR, 586

MATRIX\_VECTOR::MATRIX\_VALUES\_-  
SET, 992

MATRIX\_VALUES\_SET\_L1  
  MATRIX\_VECTOR, 587

MATRIX\_VECTOR::MATRIX\_VALUES\_-  
SET, 993

MATRIX\_VALUES\_SET\_L2  
  MATRIX\_VECTOR, 587

MATRIX\_VECTOR::MATRIX\_VALUES\_-  
SET, 993

MATRIX\_VALUES\_SET\_SP  
  MATRIX\_VECTOR, 588

MATRIX\_VECTOR::MATRIX\_VALUES\_-  
SET, 993

MATRIX\_VALUES\_SET\_SP1  
  MATRIX\_VECTOR, 588

MATRIX\_VECTOR::MATRIX\_VALUES\_-  
SET, 994

MATRIX\_VALUES\_SET\_SP2  
  MATRIX\_VECTOR, 589

MATRIX\_VECTOR::MATRIX\_VALUES\_-  
SET, 994

MATRIX\_VARIABLE\_TYPES  
  TYPES::EQUATIONS\_MAPPING\_-  
    CREATE\_VALUES\_CACHE\_TYPE,  
    1137

  TYPES::EQUATIONS\_MAPPING\_-  
    LINEAR\_TYPE, 1141

  TYPES::SOLUTION\_MAPPING\_CREATE\_-  
    VALUES\_CACHE\_TYPE, 1303

MATRIX\_VECTOR, 552

  MATRIX\_ALL\_VALUES\_SET\_DP, 560

  MATRIX\_ALL\_VALUES\_SET\_INTG, 560

  MATRIX\_ALL\_VALUES\_SET\_L, 561

  MATRIX\_ALL\_VALUES\_SET\_SP, 561

  MATRIX\_CREATE\_FINISH, 561

  MATRIX\_CREATE\_START, 562

  MATRIX\_DATA\_GET\_DP, 562

  MATRIX\_DATA\_GET\_INTG, 563

  MATRIX\_DATA\_GET\_L, 563

  MATRIX\_DATA\_GET\_SP, 563

  MATRIX\_DATA\_TYPE\_SET, 564

  MATRIX\_DESTROY, 564

  MATRIX\_DUPLICATE, 565

  MATRIX\_FINALISE, 565

  MATRIX\_INITIALISE, 566

  MATRIX\_MAX\_COLUMNS\_PER\_ROW\_-  
    GET, 566

  MATRIX\_MAX\_SIZE\_SET, 566

  MATRIX\_NUMBER\_NON\_ZEROS\_SET,  
    567

  MATRIX\_OUTPUT, 567

  MATRIX\_SIZE\_SET, 568

  MATRIX\_STORAGE\_LOCATION\_FIND,  
    568

  MATRIX\_STORAGE\_LOCATIONS\_GET,  
    569

  MATRIX\_STORAGE\_LOCATIONS\_SET,  
    570

  MATRIX\_STORAGE\_TYPE\_GET, 570

  MATRIX\_STORAGE\_TYPE\_SET, 571

  MATRIX\_VALUES\_ADD\_DP, 571

  MATRIX\_VALUES\_ADD\_DP1, 572

  MATRIX\_VALUES\_ADD\_DP2, 572

MATRIX\_VALUES\_ADD\_INTG, 573  
MATRIX\_VALUES\_ADD\_INTG1, 573  
MATRIX\_VALUES\_ADD\_INTG2, 574  
MATRIX\_VALUES\_ADD\_L, 574  
MATRIX\_VALUES\_ADD\_L1, 575  
MATRIX\_VALUES\_ADD\_L2, 575  
MATRIX\_VALUES\_ADD\_SP, 576  
MATRIX\_VALUES\_ADD\_SP1, 576  
MATRIX\_VALUES\_ADD\_SP2, 577  
MATRIX\_VALUES\_GET\_DP, 577  
MATRIX\_VALUES\_GET\_DP1, 578  
MATRIX\_VALUES\_GET\_DP2, 578  
MATRIX\_VALUES\_GET\_INTG, 579  
MATRIX\_VALUES\_GET\_INTG1, 579  
MATRIX\_VALUES\_GET\_INTG2, 580  
MATRIX\_VALUES\_GET\_L, 580  
MATRIX\_VALUES\_GET\_L1, 581  
MATRIX\_VALUES\_GET\_L2, 581  
MATRIX\_VALUES\_GET\_SP, 582  
MATRIX\_VALUES\_GET\_SP1, 582  
MATRIX\_VALUES\_GET\_SP2, 583  
MATRIX\_VALUES\_SET\_DP, 583  
MATRIX\_VALUES\_SET\_DP1, 584  
MATRIX\_VALUES\_SET\_DP2, 584  
MATRIX\_VALUES\_SET\_INTG, 585  
MATRIX\_VALUES\_SET\_INTG1, 585  
MATRIX\_VALUES\_SET\_INTG2, 586  
MATRIX\_VALUES\_SET\_L, 586  
MATRIX\_VALUES\_SET\_L1, 587  
MATRIX\_VALUES\_SET\_L2, 587  
MATRIX\_VALUES\_SET\_SP, 588  
MATRIX\_VALUES\_SET\_SP1, 588  
MATRIX\_VALUES\_SET\_SP2, 589  
MATRIX\_VECTOR\_ID, 602  
VECTOR\_ALL\_VALUES\_SET\_DP, 589  
VECTOR\_ALL\_VALUES\_SET\_INTG, 589  
VECTOR\_ALL\_VALUES\_SET\_L, 590  
VECTOR\_ALL\_VALUES\_SET\_SP, 590  
VECTOR\_CREATE\_FINISH, 591  
VECTOR\_CREATE\_START, 591  
VECTOR\_DATA\_GET\_DP, 591  
VECTOR\_DATA\_GET\_INTG, 592  
VECTOR\_DATA\_GET\_L, 592  
VECTOR\_DATA\_GET\_SP, 593  
VECTOR\_DATA\_TYPE\_SET, 593  
VECTOR\_DESTROY, 594  
VECTOR\_DUPLICATE, 594  
VECTOR\_FINALISE, 594  
VECTOR\_INITIALISE, 595  
VECTOR\_SIZE\_SET, 595  
VECTOR\_VALUES\_GET\_DP, 596  
VECTOR\_VALUES\_GET\_DP1, 596  
VECTOR\_VALUES\_GET\_INTG, 596  
VECTOR\_VALUES\_GET\_INTG1, 597  
VECTOR\_VALUES\_GET\_L, 597  
VECTOR\_VALUES\_GET\_L1, 598  
VECTOR\_VALUES\_GET\_SP, 598  
VECTOR\_VALUES\_GET\_SP1, 598  
VECTOR\_VALUES\_GET\_SP2, 598  
VECTOR\_VALUES\_SET\_DP, 599  
VECTOR\_VALUES\_SET\_DP1, 599  
VECTOR\_VALUES\_SET\_INTG, 600  
VECTOR\_VALUES\_SET\_INTG1, 600  
VECTOR\_VALUES\_SET\_L, 600  
VECTOR\_VALUES\_SET\_L1, 601  
VECTOR\_VALUES\_SET\_SP, 601  
VECTOR\_VALUES\_SET\_SP1, 602  
MATRIX\_VECTOR::DataTypes, 106  
MATRIX\_VECTOR::MATRIX\_ALL\_VALUES\_SET, 976  
MATRIX\_ALL\_VALUES\_SET\_DP, 976  
MATRIX\_ALL\_VALUES\_SET\_INTG, 976  
MATRIX\_ALL\_VALUES\_SET\_L, 976  
MATRIX\_ALL\_VALUES\_SET\_SP, 977  
MATRIX\_VECTOR::MATRIX\_DATA\_GET, 978  
MATRIX\_DATA\_GET\_DP, 978  
MATRIX\_DATA\_GET\_INTG, 978  
MATRIX\_DATA\_GET\_L, 978  
MATRIX\_DATA\_GET\_SP, 979  
MATRIX\_VECTOR::MATRIX\_VALUES\_ADD, 980  
MATRIX\_VALUES\_ADD\_DP, 980  
MATRIX\_VALUES\_ADD\_DP1, 981  
MATRIX\_VALUES\_ADD\_DP2, 981  
MATRIX\_VALUES\_ADD\_INTG, 981  
MATRIX\_VALUES\_ADD\_INTG1, 982  
MATRIX\_VALUES\_ADD\_INTG2, 982  
MATRIX\_VALUES\_ADD\_L, 982  
MATRIX\_VALUES\_ADD\_L1, 983  
MATRIX\_VALUES\_ADD\_L2, 983  
MATRIX\_VALUES\_ADD\_SP, 983  
MATRIX\_VALUES\_ADD\_SP1, 984  
MATRIX\_VALUES\_ADD\_SP2, 984  
MATRIX\_VECTOR::MATRIX\_VALUES\_GET, 985  
MATRIX\_VALUES\_GET\_DP, 985  
MATRIX\_VALUES\_GET\_DP1, 986  
MATRIX\_VALUES\_GET\_DP2, 986  
MATRIX\_VALUES\_GET\_INTG, 986  
MATRIX\_VALUES\_GET\_INTG1, 987  
MATRIX\_VALUES\_GET\_INTG2, 987  
MATRIX\_VALUES\_GET\_L, 987  
MATRIX\_VALUES\_GET\_L1, 988  
MATRIX\_VALUES\_GET\_L2, 988  
MATRIX\_VALUES\_GET\_SP, 988  
MATRIX\_VALUES\_GET\_SP1, 989  
MATRIX\_VALUES\_GET\_SP2, 989  
MATRIX\_VECTOR::MATRIX\_VALUES\_SET, 990

MATRIX\_VALUES\_SET\_DP, 990  
 MATRIX\_VALUES\_SET\_DP1, 991  
 MATRIX\_VALUES\_SET\_DP2, 991  
 MATRIX\_VALUES\_SET\_INTG, 991  
 MATRIX\_VALUES\_SET\_INTG1, 992  
 MATRIX\_VALUES\_SET\_INTG2, 992  
 MATRIX\_VALUES\_SET\_L, 992  
 MATRIX\_VALUES\_SET\_L1, 993  
 MATRIX\_VALUES\_SET\_L2, 993  
 MATRIX\_VALUES\_SET\_SP, 993  
 MATRIX\_VALUES\_SET\_SP1, 994  
 MATRIX\_VALUES\_SET\_SP2, 994  
**MATRIX\_VECTOR::StorageTypes**, 109  
**MATRIX\_VECTOR::VECTOR\_ALL\_VALUES\_SET**, 995  
 VECTOR\_ALL\_VALUES\_SET\_DP, 995  
 VECTOR\_ALL\_VALUES\_SET\_INTG, 995  
 VECTOR\_ALL\_VALUES\_SET\_L, 995  
 VECTOR\_ALL\_VALUES\_SET\_SP, 996  
**MATRIX\_VECTOR::VECTOR\_DATA\_GET**, 997  
 VECTOR\_DATA\_GET\_DP, 997  
 VECTOR\_DATA\_GET\_INTG, 997  
 VECTOR\_DATA\_GET\_L, 997  
 VECTOR\_DATA\_GET\_SP, 998  
**MATRIX\_VECTOR::VECTOR\_VALUES\_GET**, 999  
 VECTOR\_VALUES\_GET\_DP, 999  
 VECTOR\_VALUES\_GET\_DP1, 999  
 VECTOR\_VALUES\_GET\_INTG, 999  
 VECTOR\_VALUES\_GET\_INTG1, 1000  
 VECTOR\_VALUES\_GET\_L, 1000  
 VECTOR\_VALUES\_GET\_L1, 1000  
 VECTOR\_VALUES\_GET\_SP, 1001  
 VECTOR\_VALUES\_GET\_SP1, 1001  
**MATRIX\_VECTOR::VECTOR\_VALUES\_SET**, 1002  
 VECTOR\_VALUES\_SET\_DP, 1002  
 VECTOR\_VALUES\_SET\_DP1, 1002  
 VECTOR\_VALUES\_SET\_INTG, 1002  
 VECTOR\_VALUES\_SET\_INTG1, 1003  
 VECTOR\_VALUES\_SET\_L, 1003  
 VECTOR\_VALUES\_SET\_L1, 1003  
 VECTOR\_VALUES\_SET\_SP, 1004  
 VECTOR\_VALUES\_SET\_SP1, 1004  
**MATRIX\_VECTOR::DataTypes**  
 MATRIX\_VECTOR\_DP\_TYPE, 106  
 MATRIX\_VECTOR\_INTG\_TYPE, 107  
 MATRIX\_VECTOR\_L\_TYPE, 107  
 MATRIX\_VECTOR\_SP\_TYPE, 108  
**MATRIX\_VECTOR\_DP\_TYPE**  
 MATRIX\_VECTOR\_DataTypes, 106  
**MATRIX\_VECTOR\_ID**  
 MATRIX\_VECTOR, 602  
**MATRIX\_VECTOR\_INTG\_TYPE**

MATRIX\_VECTOR\_DataTypes, 107  
**MATRIX\_VECTOR\_L\_TYPE**  
 MATRIX\_VECTOR\_DataTypes, 107  
**MATRIX\_VECTOR\_SP\_TYPE**  
 MATRIX\_VECTOR\_DataTypes, 108  
**MATRIX\_VECTOR\_StorageTypes**  
 MATRIX\_BLOCK\_STORAGE\_TYPE, 109  
 MATRIX\_COLUMN\_MAJOR\_STORAGE\_TYPE, 110  
 MATRIX\_COMPRESSED\_COLUMN\_STORAGE\_TYPE, 110  
 MATRIX\_COMPRESSED\_ROW\_STORAGE\_TYPE, 110  
 MATRIX\_DIAGONAL\_STORAGE\_TYPE, 111  
 MATRIX\_ROW\_COLUMN\_STORAGE\_TYPE, 111  
 MATRIX\_ROW\_MAJOR\_STORAGE\_TYPE, 111  
**MatSetLocalToGlobalMapping**  
 CMISS\_PETSC::interface, 804  
**MatSetOption**  
 CMISS\_PETSC::interface, 804  
**MatSetSizes**  
 CMISS\_PETSC::interface, 804  
**MatSetValue**  
 CMISS\_PETSC::interface, 804  
**MatSetValueLocal**  
 CMISS\_PETSC::interface, 805  
**MatSetValues**  
 CMISS\_PETSC::interface, 805  
**MatSetValuesLocal**  
 CMISS\_PETSC::interface, 805  
**MatView**  
 CMISS\_PETSC::interface, 805  
**MatZeroEntries**  
 CMISS\_PETSC::interface, 805  
**MAX\_M**  
 TYPES::MATRIX\_TYPE, 1254  
**MAX\_N**  
 TYPES::MATRIX\_TYPE, 1254  
**MAX\_NUM\_BINARY\_FILES**  
 BINARY\_FILE, 226  
**MAX\_NUMBER\_OF\_COLUMNS**  
 TYPES::ELEMENT\_MATRIX\_TYPE, 1122  
**MAX\_NUMBER\_OF\_DERIVATIVES**  
 TYPES::FIELD\_PARAM\_TO\_DOF\_MAP\_TYPE, 1214  
 TYPES::FIELD\_SCALING\_TYPE, 1222  
**MAX\_NUMBER\_OF\_ELEMENT\_PARAMETERS**  
 TYPES::FIELD\_SCALING\_TYPE, 1222  
**MAX\_NUMBER\_OF\_INTERPOLATION\_PARAMETERS**

TYPES::FIELD\_VARIABLE\_-  
COMPONENT\_TYPE, 1230  
TYPES::FIELD\_VARIABLE\_TYPE, 1234  
MAX\_NUMBER\_OF\_ROWS  
TYPES::ELEMENT\_MATRIX\_TYPE, 1122  
TYPES::ELEMENT\_VECTOR\_TYPE, 1124  
MAX\_OUTPUT\_LINES  
BASE\_ROUTINES, 210  
MAX\_PARTIAL\_DERIVATIVE\_INDEX  
TYPES::FIELD\_INTERPOLATED\_POINT\_-  
TYPE, 1208  
MAXBINFILIES  
binary\_file\_c.c, 1358  
MAXIMUM\_COLUMN\_INDICES\_PER\_ROW  
TYPES::DISTRIBUTED\_MATRIX\_-  
PETSC\_TYPE, 1071  
TYPES::MATRIX\_TYPE, 1254  
MAXIMUM\_EXTENT  
TYPES::GENERATED\_MESH\_-  
REGULAR\_TYPE, 1238  
MAXIMUM\_NUMBER\_OF\_DERIVATIVES  
TYPES::BASIS\_TYPE, 1048  
TYPES::DOMAIN\_NODES\_TYPE, 1114  
MAXIMUM\_NUMBER\_OF\_ELEMENT\_-  
PARAMETERS  
TYPES::DOMAIN\_ELEMENTS\_TYPE,  
1093  
MAXIMUM\_NUMBER\_OF\_FUNCTION\_-  
EVALUATIONS  
TYPES::NONLINEAR\_SOLVER\_TYPE,  
1281  
MAXIMUM\_NUMBER\_OF\_ITERATIONS  
TYPES::LINEAR\_ITERATIVE\_SOLVER\_-  
TYPE, 1248  
TYPES::NONLINEAR\_SOLVER\_TYPE,  
1281  
MESH  
TYPES::DECOMPOSITION\_TYPE, 1066  
TYPES::DECOMPOSITIONS\_TYPE, 1067  
TYPES::DOMAIN\_TYPE, 1120  
TYPES::GENERATED\_MESH\_TYPE, 1240  
TYPES::MESH\_DOFS\_TYPE, 1256  
TYPES::MESH\_ELEMENTS\_TYPE, 1259  
TYPES::MESH\_NODES\_TYPE, 1263  
TYPES::MESH\_TOPOLOGY\_TYPE, 1266  
MESH\_COMPONENT\_NUMBER  
TYPES::DECOMPOSITION\_TYPE, 1066  
TYPES::DOMAIN\_TYPE, 1120  
TYPES::FIELD\_CREATE\_VALUES\_-  
CACHE\_TYPE, 1197  
TYPES::FIELD\_SCALING\_TYPE, 1222  
TYPES::FIELD\_VARIABLE\_-  
COMPONENT\_TYPE, 1230  
TYPES::MESH\_TOPOLOGY\_TYPE, 1266  
MESH\_CREATE\_FINISH  
MESH\_ROUTINES, 622  
MESH\_CREATE\_START  
MESH\_ROUTINES, 623  
MESH\_DESTROY  
MESH\_ROUTINES, 624  
MESH\_DIMENSION  
TYPES::GENERATED\_MESH\_-  
REGULAR\_TYPE, 1239  
MESH\_EMBEDDED  
TYPES::MESH\_TYPE, 1269  
MESH\_FINALISE  
MESH\_ROUTINES, 624  
MESH\_FINISHED  
TYPES::MESH\_TYPE, 1270  
MESH\_INITIALISE  
MESH\_ROUTINES, 624  
MESH\_NUMBER\_OF\_COMPONENTS\_GET  
MESH\_ROUTINES, 625  
MESH\_NUMBER\_OF\_COMPONENTS\_SET\_-  
NUMBER  
MESH\_ROUTINES, 625  
MESH\_ROUTINES::MESH\_NUMBER\_-  
OF\_COMPONENTS\_SET, 1005  
MESH\_NUMBER\_OF\_COMPONENTS\_SET\_-  
PTR  
MESH\_ROUTINES, 626  
MESH\_ROUTINES::MESH\_NUMBER\_-  
OF\_COMPONENTS\_SET, 1005  
MESH\_NUMBER\_OF\_ELEMENTS\_GET  
MESH\_ROUTINES, 626  
MESH\_NUMBER\_OF\_ELEMENTS\_SET\_-  
NUMBER  
MESH\_ROUTINES, 626  
MESH\_ROUTINES::MESH\_NUMBER\_-  
OF\_ELEMENTS\_SET, 1006  
MESH\_NUMBER\_OF\_ELEMENTS\_SET\_PTR  
MESH\_ROUTINES, 627  
MESH\_ROUTINES::MESH\_NUMBER\_-  
OF\_ELEMENTS\_SET, 1006  
MESH\_ROUTINES, 603  
DECOMPOSITION\_CREATE\_FINISH, 608  
DECOMPOSITION\_CREATE\_START, 609  
DECOMPOSITION\_DESTROY, 609  
DECOMPOSITION\_ELEMENT\_-  
DOMAIN\_CALCULATE, 610  
DECOMPOSITION\_ELEMENT\_-  
DOMAIN\_GET, 610  
DECOMPOSITION\_ELEMENT\_-  
DOMAIN\_SET, 611  
DECOMPOSITION\_MESH\_-  
COMPONENT\_NUMBER\_GET, 611  
DECOMPOSITION\_MESH\_-  
COMPONENT\_NUMBER\_SET, 611

DECOMPOSITION\_NUMBER\_OF\_-  
     DOMAINS\_GET, 612  
 DECOMPOSITION\_NUMBER\_OF\_-  
     DOMAINS\_SET, 612  
 DOMAIN\_MAPPINGS\_NODES\_FINALISE,  
     613  
 DOMAIN\_MAPPINGS\_NODES\_-  
     INITIALISE, 613  
 DOMAIN\_TOPOLOGY\_CALCULATE, 614  
 DOMAIN\_TOPOLOGY\_DOFS\_FINALISE,  
     614  
 DOMAIN\_TOPOLOGY\_DOFS\_-  
     INITIALISE, 615  
 DOMAIN\_TOPOLOGY\_ELEMENT\_-  
     FINALISE, 615  
 DOMAIN\_TOPOLOGY\_ELEMENT\_-  
     INITIALISE, 616  
 DOMAIN\_TOPOLOGY\_ELEMENTS\_-  
     FINALISE, 616  
 DOMAIN\_TOPOLOGY\_ELEMENTS\_-  
     INITIALISE, 617  
 DOMAIN\_TOPOLOGY\_FINALISE, 617  
 DOMAIN\_TOPOLOGY\_INITIALISE, 618  
 DOMAIN\_TOPOLOGY\_INITIALISE\_-  
     FROM\_MESH, 618  
 DOMAIN\_TOPOLOGY\_LINE\_FINALISE,  
     619  
 DOMAIN\_TOPOLOGY\_LINE\_INITIALISE,  
     619  
 DOMAIN\_TOPOLOGY\_LINES\_FINALISE,  
     619  
 DOMAIN\_TOPOLOGY\_LINES\_-  
     INITIALISE, 620  
 DOMAIN\_TOPOLOGY\_NODE\_FINALISE,  
     620  
 DOMAIN\_TOPOLOGY\_NODE\_-  
     INITIALISE, 621  
 DOMAIN\_TOPOLOGY\_NODES\_-  
     FINALISE, 621  
 DOMAIN\_TOPOLOGY\_NODES\_-  
     INITIALISE, 622  
 DOMAIN\_TOPOLOGY\_NODES\_-  
     SURROUNDING\_ELEMENTS\_-  
         CALCULATE, 622  
 MESH\_CREATE\_FINISH, 622  
 MESH\_CREATE\_START, 623  
 MESH\_DESTROY, 624  
 MESH\_FINALISE, 624  
 MESH\_INITIALISE, 624  
 MESH\_NUMBER\_OF\_COMPONENTS\_-  
     GET, 625  
 MESH\_NUMBER\_OF\_COMPONENTS\_-  
     SET\_NUMBER, 625  
 MESH\_NUMBER\_OF\_COMPONENTS\_-  
     SET\_PTR, 626  
 MESH\_NUMBER\_OF\_ELEMENTS\_GET,  
     626  
 MESH\_NUMBER\_OF\_ELEMENTS\_SET\_-  
     NUMBER, 626  
 MESH\_NUMBER\_OF\_ELEMENTS\_SET\_-  
     PTR, 627  
 MESH\_TOPOLOGY\_CALCULATE, 627  
 MESH\_TOPOLOGY\_DOFS\_CALCULATE,  
     628  
 MESH\_TOPOLOGY\_DOFS\_FINALISE, 628  
 MESH\_TOPOLOGY\_DOFS\_INITIALISE,  
     629  
 MESH\_TOPOLOGY\_ELEMENT\_-  
     FINALISE, 629  
 MESH\_TOPOLOGY\_ELEMENT\_-  
     INITIALISE, 630  
 MESH\_TOPOLOGY\_ELEMENTS\_-  
     ADJACENT\_ELEMENTS\_-  
         CALCULATE, 630  
 MESH\_TOPOLOGY\_ELEMENTS\_BASIS\_-  
     SET, 630  
 MESH\_TOPOLOGY\_ELEMENTS\_-  
     CREATE\_FINISH, 631  
 MESH\_TOPOLOGY\_ELEMENTS\_-  
     CREATE\_START, 631  
 MESH\_TOPOLOGY\_ELEMENTS\_-  
     DESTROY, 632  
 MESH\_TOPOLOGY\_ELEMENTS\_-  
     ELEMENT\_BASIS\_GET, 632  
 MESH\_TOPOLOGY\_ELEMENTS\_-  
     ELEMENT\_BASIS\_SET, 633  
 MESH\_TOPOLOGY\_ELEMENTS\_-  
     ELEMENT\_NODES\_GET, 633  
 MESH\_TOPOLOGY\_ELEMENTS\_-  
     ELEMENT\_NODES\_SET, 634  
 MESH\_TOPOLOGY\_ELEMENTS\_-  
     FINALISE, 635  
 MESH\_TOPOLOGY\_ELEMENTS\_-  
     INITIALISE, 635  
 MESH\_TOPOLOGY\_ELEMENTS\_-  
     NUMBER\_GET, 636  
 MESH\_TOPOLOGY\_ELEMENTS\_-  
     NUMBER\_SET, 636  
 MESH\_TOPOLOGY\_FINALISE, 637  
 MESH\_TOPOLOGY\_INITIALISE, 637  
 MESH\_TOPOLOGY\_NODE\_FINALISE,  
     638  
 MESH\_TOPOLOGY\_NODE\_INITIALISE,  
     638  
 MESH\_TOPOLOGY\_NODES\_-  
     CALCULATE, 638

MESH\_TOPOLOGY\_NODES\_-  
  DERIVATIVES\_CALCULATE, 639  
MESH\_TOPOLOGY\_NODES\_FINALISE,  
  639  
MESH\_TOPOLOGY\_NODES\_INITIALISE,  
  640  
MESH\_TOPOLOGY\_NODES\_-  
  SURROUNDING\_ELEMENTS\_-  
  CALCULATE, 640  
MESH\_USER\_NUMBER\_FIND, 641  
MESSES\_FINALISE, 641  
MESSES\_INITIALISE, 642  
MESH\_ROUTINES::DecompositionTypes, 113  
MESH\_ROUTINES::MESH\_NUMBER\_OF\_-  
  COMPONENTS\_SET, 1005  
  MESH\_NUMBER\_OF\_COMPONENTS\_-  
    SET\_NUMBER, 1005  
  MESH\_NUMBER\_OF\_COMPONENTS\_-  
    SET\_PTR, 1005  
MESH\_ROUTINES::MESH\_NUMBER\_OF\_-  
  ELEMENTS\_SET, 1006  
  MESH\_NUMBER\_OF\_ELEMENTS\_SET\_-  
    NUMBER, 1006  
  MESH\_NUMBER\_OF\_ELEMENTS\_SET\_-  
    PTR, 1006  
MESH\_ROUTINES\_DecompositionTypes  
  DECOMPOSITION\_ALL\_TYPE, 113  
  DECOMPOSITION\_CALCULATED\_TYPE,  
    113  
  DECOMPOSITION\_USER\_DEFINED\_-  
    TYPE, 114  
MESH\_TOPOLOGY\_CALCULATE  
  MESH\_ROUTINES, 627  
MESH\_TOPOLOGY\_DOFS\_CALCULATE  
  MESH\_ROUTINES, 628  
MESH\_TOPOLOGY\_DOFS\_FINALISE  
  MESH\_ROUTINES, 628  
MESH\_TOPOLOGY\_DOFS\_INITIALISE  
  MESH\_ROUTINES, 629  
MESH\_TOPOLOGY\_ELEMENT\_FINALISE  
  MESH\_ROUTINES, 629  
MESH\_TOPOLOGY\_ELEMENT\_INITIALISE  
  MESH\_ROUTINES, 630  
MESH\_TOPOLOGY\_ELEMENTS\_-  
  ADJACENT\_ELEMENTS\_-  
  CALCULATE  
  MESH\_ROUTINES, 630  
MESH\_TOPOLOGY\_ELEMENTS\_BASIS\_SET  
  MESH\_ROUTINES, 630  
MESH\_TOPOLOGY\_ELEMENTS\_CREATE\_-  
  FINISH  
  MESH\_ROUTINES, 631  
MESH\_TOPOLOGY\_ELEMENTS\_CREATE\_-  
  START  
  MESH\_ROUTINES, 631  
MESH\_TOPOLOGY\_ELEMENTS\_DESTROY  
  MESH\_ROUTINES, 632  
MESH\_TOPOLOGY\_ELEMENTS\_ELEMENT\_-  
  BASIS\_GET  
  MESH\_ROUTINES, 632  
MESH\_TOPOLOGY\_ELEMENTS\_ELEMENT\_-  
  BASIS\_SET  
  MESH\_ROUTINES, 633  
MESH\_TOPOLOGY\_ELEMENTS\_ELEMENT\_-  
  NODES\_GET  
  MESH\_ROUTINES, 633  
MESH\_TOPOLOGY\_ELEMENTS\_ELEMENT\_-  
  NODES\_SET  
  MESH\_ROUTINES, 634  
MESH\_TOPOLOGY\_ELEMENTS\_FINALISE  
  MESH\_ROUTINES, 635  
MESH\_TOPOLOGY\_ELEMENTS\_INITIALISE  
  MESH\_ROUTINES, 635  
MESH\_TOPOLOGY\_ELEMENTS\_NUMBER\_-  
  GET  
  MESH\_ROUTINES, 636  
MESH\_TOPOLOGY\_ELEMENTS\_NUMBER\_-  
  SET  
  MESH\_ROUTINES, 636  
MESH\_TOPOLOGY\_FINALISE  
  MESH\_ROUTINES, 637  
MESH\_TOPOLOGY\_INITIALISE  
  MESH\_ROUTINES, 637  
MESH\_TOPOLOGY\_NODE\_FINALISE  
  MESH\_ROUTINES, 638  
MESH\_TOPOLOGY\_NODE\_INITIALISE  
  MESH\_ROUTINES, 638  
MESH\_TOPOLOGY\_NODES\_CALCULATE  
  MESH\_ROUTINES, 638  
MESH\_TOPOLOGY\_NODES\_DERIVATIVES\_-  
  CALCULATE  
  MESH\_ROUTINES, 639  
MESH\_TOPOLOGY\_NODES\_FINALISE  
  MESH\_ROUTINES, 639  
MESH\_TOPOLOGY\_NODES\_INITIALISE  
  MESH\_ROUTINES, 640  
MESH\_TOPOLOGY\_NODES\_-  
  SURROUNDING\_ELEMENTS\_-  
  CALCULATE  
  MESH\_ROUTINES, 640  
MESH\_USER\_NUMBER\_FIND  
  MESH\_ROUTINES, 641  
MESSES  
  TYPES::MESH\_TYPE, 1270  
  TYPES::MESSES\_TYPE, 1272  
  TYPES::REGION\_TYPE, 1301  
MESSES\_FINALISE  
  MESH\_ROUTINES, 641

MESHERS\_INITIALISE  
    MESH\_ROUTINES, 642

MPI\_COMM  
    COMP\_ENVIRONMENT::COMPUTATIONAL\_-  
        ENVIRONMENT\_TYPE, 822

MPI\_COMPUTATIONAL\_NODE\_TYPE\_DATA  
    COMP\_ENVIRONMENT, 299

MPI\_ERROR\_CHECK  
    CMISS\_MPI, 236

MPI\_RECEIVE\_REQUEST  
    TYPES::DISTRIBUTED\_VECTOR\_-  
        TRANSFER\_TYPE, 1082

MPI\_SEND\_REQUEST  
    TYPES::DISTRIBUTED\_VECTOR\_-  
        TRANSFER\_TYPE, 1082

MPI\_TYPE  
    COMP\_ENVIRONMENT::MPI\_-  
        COMPUTATIONAL\_NODE\_TYPE,  
            826

MY\_COMPUTATIONAL\_NODE\_NUMBER  
    COMP\_ENVIRONMENT::COMPUTATIONAL\_-  
        ENVIRONMENT\_TYPE, 822

N

    TYPES::DISTRIBUTED\_MATRIX\_-  
        PETSC\_TYPE, 1071

    TYPES::DISTRIBUTED\_VECTOR\_-  
        CMISS\_TYPE, 1077

    TYPES::DISTRIBUTED\_VECTOR\_-  
        PETSC\_TYPE, 1080

    TYPES::MATRIX\_TYPE, 1254

    TYPES::VECTOR\_TYPE, 1342

NAME

    BASE\_ROUTINES::ROUTINE\_LIST\_-  
        ITEM\_TYPE, 773

    BASE\_ROUTINES::ROUTINE\_STACK\_-  
        ITEM\_TYPE, 777

NEXT\_ROUTINE

    BASE\_ROUTINES::ROUTINE\_LIST\_-  
        ITEM\_TYPE, 773

NIL

    TREES::TREE\_TYPE, 1038

NODAL\_INFO\_SET

    FIELD\_IO\_ROUTINES::FIELD\_IO\_-  
        NODAL\_INFO\_SET, 848

NODE\_AT\_COLLAPSE  
    TYPES::BASIS\_TYPE, 1048

NODE\_CHECK\_EXISTS  
    NODE\_ROUTINES, 643

NODE\_DESTROY  
    NODE\_ROUTINES, 643

NODE\_DOF2PARAM\_MAP  
    TYPES::FIELD\_DOF\_TO\_PARAM\_MAP\_-  
        TYPE, 1200

    NODE\_DOMAIN  
        TYPES::DOMAIN\_TYPE, 1120

    NODE\_INITIAL\_POSITION\_SET  
        NODE\_ROUTINES, 644

    NODE\_LINES  
        TYPES::DOMAIN\_NODE\_TYPE, 1112

    NODE\_NAME  
        COMP\_ENVIRONMENT::COMPUTATIONAL\_-  
            NODE\_TYPE, 824

    NODE\_NAME\_LENGTH  
        COMP\_ENVIRONMENT::COMPUTATIONAL\_-  
            NODE\_TYPE, 824

    NODE\_NUMBER\_SET  
        NODE\_ROUTINES, 644

    NODE\_NUMBERS\_IN\_LOCAL\_LINE  
        TYPES::BASIS\_TYPE, 1048

    NODE\_PARAM2DOF\_MAP  
        TYPES::FIELD\_PARAM\_TO\_DOF\_MAP\_-  
            TYPE, 1214

    NODE\_POSITION\_INDEX  
        TYPES::BASIS\_TYPE, 1048

    NODE\_POSITION\_INDEX\_INV  
        TYPES::BASIS\_TYPE, 1048

    NODE\_ROUTINES, 643

        NODE\_CHECK\_EXISTS, 643

        NODE\_DESTROY, 643

        NODE\_INITIAL\_POSITION\_SET, 644

        NODE\_NUMBER\_SET, 644

        NODES\_CREATE\_FINISH, 644

        NODES\_CREATE\_START, 644

        NODES\_FINALISE, 645

        NODES\_INITIALISE, 645

    NODE\_TREE  
        TYPES::NODES\_TYPE, 1275

    NODES

        TYPES::DOMAIN\_MAPPINGS\_TYPE, 1109

        TYPES::DOMAIN\_NODES\_TYPE, 1114

        TYPES::DOMAIN\_TOPOLOGY\_TYPE,  
            1118

        TYPES::MESH\_NODES\_TYPE, 1263

        TYPES::MESH\_TOPOLOGY\_TYPE, 1267

        TYPES::NODES\_TYPE, 1275

        TYPES::REGION\_TYPE, 1301

    NODES\_CREATE\_FINISH  
        NODE\_ROUTINES, 644

    NODES\_CREATE\_START  
        NODE\_ROUTINES, 644

    NODES\_FINALISE  
        NODE\_ROUTINES, 645

    NODES\_FINISHED  
        TYPES::NODES\_TYPE, 1275

    NODES\_IN\_FACE  
        TYPES::DOMAIN\_FACE\_TYPE, 1096

    NODES\_IN\_LINE

TYPES::DOMAIN\_LINE\_TYPE, 1102  
NODES\_INITIALISE  
  NODE\_ROUTINES, 645  
NONLINEAR\_DATA  
  TYPES::EQUATIONS\_TYPE, 1195  
NONLINEAR\_JACOBIAN\_TYPE  
  TYPES::EQUATIONS\_TYPE, 1195  
NONLINEAR\_MAPPING  
  TYPES::EQUATIONS\_MAPPING\_TYPE,  
    1150  
NONLINEAR\_MATRICES  
  TYPES::EQUATIONS\_JACOBIAN\_TYPE,  
    1134  
  TYPES::EQUATIONS\_MATRICES\_TYPE,  
    1160  
NONLINEAR\_SOLVE\_TYPE  
  TYPES::NONLINEAR\_SOLVER\_TYPE,  
    1281  
NONLINEAR\_SOLVER  
  TYPES::NONLINEAR\_LINESearch\_-  
    SOLVER\_TYPE, 1279  
  TYPES::NONLINEAR\_TRUSTREGION\_-  
    SOLVER\_TYPE, 1283  
  TYPES::SOLVER\_TYPE, 1331  
NORMALISE\_DP  
  MATHS, 551  
  MATHS::NORMALISE, 974  
NORMALISE\_SP  
  MATHS, 551  
  MATHS::NORMALISE, 974  
NUM\_BLOCKS  
  COMP\_ENVIRONMENT::MPI\_-  
    COMPUTATIONAL\_NODE\_TYPE,  
      826  
NUM\_BYTES  
  BINARY\_FILE::BINARY\_TAG\_TYPE, 784  
NUM\_HEADER\_BYTES  
  BINARY\_FILE::BINARY\_TAG\_TYPE, 784  
NUM\_SUBTAGS  
  BINARY\_FILE::BINARY\_TAG\_TYPE, 784  
NUMBER  
  TYPES::DECOMPOSITION\_LINE\_TYPE,  
    1059  
  TYPES::DOMAIN\_ELEMENT\_TYPE, 1091  
  TYPES::DOMAIN\_FACE\_TYPE, 1097  
  TYPES::DOMAIN\_LINE\_TYPE, 1102  
NUMBER\_BASIS\_FUNCTIONS  
  TYPES::BASIS\_FUNCTIONS\_TYPE, 1040  
NUMBER\_COMPUTATIONAL\_NODES  
  COMP\_ENVIRONMENT::COMPUTATIONAL\_-  
    ENVIRONMENT\_TYPE, 823  
NUMBER\_IN\_LIST  
  LISTS::LIST\_TYPE, 958  
NUMBER\_IN\_TREE  
  TREES::TREE\_TYPE, 1038  
NUMBER\_LEVELS  
  COMP\_ENVIRONMENT::CACHE\_TYPE,  
    821  
NUMBER\_NON\_ZEROS  
  TYPES::DISTRIBUTED\_MATRIX\_-  
    PETSC\_TYPE, 1072  
  TYPES::MATRIX\_TYPE, 1255  
NUMBER\_OF\_ADJACENT\_DOMAINS  
  TYPES::DOMAIN\_MAPPING\_TYPE, 1107  
NUMBER\_OF\_ADJACENT\_ELEMENTS  
  TYPES::DECOMPOSITION\_ELEMENT\_-  
    TYPE, 1056  
  TYPES::MESH\_ELEMENT\_TYPE, 1258  
NUMBER\_OF AREAS  
  TYPES::FIELD\_GEOMETRIC\_-  
    PARAMETERS\_TYPE, 1204  
NUMBER\_OF\_BOUNDARY  
  TYPES::DOMAIN\_MAPPING\_TYPE, 1107  
NUMBER\_OF\_COLLAPSED\_XI  
  TYPES::BASIS\_TYPE, 1049  
NUMBER\_OF\_COLUMNS  
  TYPES::ELEMENT\_MATRIX\_TYPE, 1122  
  TYPES::EQUATIONS\_JACOBIAN\_TO\_-  
    VARIABLE\_MAP\_TYPE, 1132  
  TYPES::EQUATIONS\_JACOBIAN\_TYPE,  
    1134  
  TYPES::EQUATIONS\_MATRIX\_TO\_-  
    VARIABLE\_MAP\_TYPE, 1164  
  TYPES::EQUATIONS\_MATRIX\_TYPE,  
    1166  
  TYPES::SOLVER\_COL\_TO\_EQUATIONS\_-  
    SETS\_MAP\_TYPE, 1316  
  TYPES::SOLVER\_JACOBIAN\_TYPE, 1320  
  TYPES::SOLVER\_MATRIX\_TYPE, 1326  
NUMBER\_OF\_COMPONENTS  
  FIELD\_IO\_ROUTINES::FIELD\_IO\_INFO\_-  
    SET, 847  
  TYPES::FIELD\_CREATE\_VALUES\_-  
    CACHE\_TYPE, 1197  
  TYPES::FIELD\_VARIABLE\_TYPE, 1234  
  TYPES::MESH\_TYPE, 1270  
NUMBER\_OF\_CONSTANT\_DOFS  
  TYPES::FIELD\_DOF\_TO\_PARAM\_MAP\_-  
    TYPE, 1201  
NUMBER\_OF\_CONSTANT\_PARAMETERS  
  TYPES::FIELD\_PARAM\_TO\_DOF\_MAP\_-  
    TYPE, 1214  
NUMBER\_OF\_COORDINATE\_SYSTEMS  
  COORDINATE\_-  
    ROUTINES::COORDINATE\_-  
      SYSTEMS\_TYPE, 840  
NUMBER\_OF\_DECOMPOSITIONS  
  TYPES::DECOMPOSITIONS\_TYPE, 1067

NUMBER\_OF\_DERIVATIVES  
 TYPES::BASIS\_TYPE, 1049  
 TYPES::DOMAIN\_NODE\_TYPE, 1112  
 TYPES::MESH\_NODE\_TYPE, 1261

NUMBER\_OF\_DIMENSIONS  
 TYPES::COORDINATE\_SYSTEM\_TYPE,  
 1053  
 TYPES::DOMAIN\_TYPE, 1120  
 TYPES::MESH\_TYPE, 1270

NUMBER\_OF\_DOFS  
 TYPES::DOMAIN\_DOFS\_TYPE, 1089  
 TYPES::FIELD\_DOF\_TO\_PARAM\_MAP\_-  
 TYPE, 1201  
 TYPES::FIELD\_VARIABLE\_TYPE, 1235  
 TYPES::MESH\_DOFS\_TYPE, 1256  
 TYPES::SOLVER\_COL\_TO\_EQUATIONS\_-  
 SETS\_MAP\_TYPE, 1316

NUMBER\_OF\_DOMAIN\_LOCAL  
 TYPES::DOMAIN\_MAPPING\_TYPE, 1108

NUMBER\_OF\_DOMAINS  
 TYPES::DECOMPOSITION\_TYPE, 1066  
 TYPES::DOMAIN\_GLOBAL\_MAPPING\_-  
 TYPE, 1100  
 TYPES::DOMAIN\_MAPPING\_TYPE, 1108

NUMBER\_OF\_EDGES\_CUT  
 TYPES::DECOMPOSITION\_TYPE, 1066

NUMBER\_OF\_ELEMENT\_DOFS  
 TYPES::FIELD\_DOF\_TO\_PARAM\_MAP\_-  
 TYPE, 1201

NUMBER\_OF\_ELEMENT\_PARAMETERS  
 TYPES::BASIS\_TYPE, 1049  
 TYPES::FIELD\_PARAM\_TO\_DOF\_MAP\_-  
 TYPE, 1214

NUMBER\_OF\_ELEMENTS  
 FIELD\_IO\_ROUTINES::FIELD\_IO\_-  
 ELEMENTALL\_INFO\_SET, 846  
 TYPES::DOMAIN\_ELEMENTS\_TYPE,  
 1094  
 TYPES::MESH\_ELEMENTS\_TYPE, 1260  
 TYPES::MESH\_TYPE, 1270

NUMBER\_OF\_ELEMENTS\_XI  
 TYPES::GENERATED\_MESH\_-  
 REGULAR\_TYPE, 1239

NUMBER\_OF\_EMBEDDED\_MESHES  
 TYPES::MESH\_TYPE, 1270

NUMBER\_OF\_EQUATIONS\_SET\_-  
 LINEARITY\_TYPES  
 EQUATIONS\_SET\_CONSTANTS\_-  
 LinearityTypes, 49

NUMBER\_OF\_EQUATIONS\_SET\_SOLUTION\_-  
 METHODS  
 EQUATIONS\_SET\_CONSTANTS\_-  
 SolutionMethods, 56

NUMBER\_OF\_EQUATIONS\_SET\_TIME\_-  
 TYPES  
 EQUATIONS\_SET\_CONSTANTS\_-  
 TimeDependenceTypes, 53

NUMBER\_OF\_EQUATIONS\_SETS  
 TYPES::EQUATIONS\_SETS\_TYPE, 1183  
 TYPES::SOLUTION\_MAPPING\_TYPE,  
 1306

TYPES::SOLVER\_DOF\_TO\_VARIABLE\_-  
 MAP\_TYPE, 1318

NUMBER\_OF\_FACES  
 TYPES::DOMAIN\_FACES\_TYPE, 1098  
 TYPES::MESH\_TYPE, 1270

NUMBER\_OF\_FIELDS  
 TYPES::FIELDS\_TYPE, 1236

NUMBER\_OF\_FIELDS\_USING  
 TYPES::FIELD\_GEOMETRIC\_-  
 PARAMETERS\_TYPE, 1204

NUMBER\_OF\_GAUSS  
 TYPES::QUADRATURE\_SCHEME\_TYPE,  
 1296

NUMBER\_OF\_GAUSS\_XI  
 TYPES::QUADRATURE\_TYPE, 1298

NUMBER\_OF\_GENERATED\_MESHES  
 TYPES::GENERATED\_MESHES\_TYPE,  
 1242

NUMBER\_OF\_GHOST  
 TYPES::DOMAIN\_MAPPING\_TYPE, 1108

NUMBER\_OF\_GLOBAL  
 TYPES::DOMAIN\_MAPPING\_TYPE, 1108

NUMBER\_OF\_GLOBAL\_DOFS  
 TYPES::FIELD\_VARIABLE\_TYPE, 1235  
 TYPES::SOLVER\_COL\_TO\_EQUATIONS\_-  
 SETS\_MAP\_TYPE, 1316

NUMBER\_OF\_GLOBAL\_ROWS  
 TYPES::EQUATIONS\_MAPPING\_TYPE,  
 1150  
 TYPES::EQUATIONS\_MATRICES\_TYPE,  
 1160  
 TYPES::SOLUTION\_MAPPING\_TYPE,  
 1306  
 TYPES::SOLVER\_MATRICES\_TYPE, 1323

NUMBER\_OF\_INTERNAL  
 TYPES::DOMAIN\_MAPPING\_TYPE, 1108

NUMBER\_OF\_INVOCATIONS  
 BASE\_ROUTINES::ROUTINE\_LIST\_-  
 ITEM\_TYPE, 774

NUMBER\_OF\_ITERATIONS  
 TYPES::PROBLEM\_NONLINEAR\_DATA\_-  
 TYPE, 1287

NUMBER\_OF\_LINEAR\_EQUATIONS\_-  
 MATRICES  
 TYPES::EQUATIONS\_MAPPING\_-  
 CREATE\_VALUES\_CACHE\_TYPE,

1138  
TYPES::EQUATIONS\_MAPPING\_-  
LINEAR\_TYPE, 1141  
TYPES::EQUATIONS\_TO\_SOLVER\_-  
MATRIX\_MAPS\_SM\_TYPE, 1192  
TYPES::SOLVER\_COL\_TO\_EQUATIONS\_-  
MAP\_TYPE, 1313  
TYPES::VARIABLE\_TO\_EQUATIONS\_-  
MATRICES\_MAP\_TYPE, 1338  
NUMBER\_OF\_LINEAR\_MATRICES  
TYPES::EQUATIONS\_MATRICES\_-  
LINEAR\_TYPE, 1152  
NUMBER\_OF\_LINEAR\_MATRIX\_VARIABLES  
TYPES::EQUATIONS\_MAPPING\_-  
LINEAR\_TYPE, 1141  
NUMBER\_OF\_LINES  
TYPES::DECOMPOSITION\_LINES\_TYPE,  
1061  
TYPES::DOMAIN\_LINES\_TYPE, 1104  
TYPES::FIELD\_GEOMETRIC\_-  
PARAMETERS\_TYPE, 1204  
TYPES::MESH\_TYPE, 1270  
NUMBER\_OF\_LOCAL  
TYPES::DOMAIN\_MAPPING\_TYPE, 1108  
NUMBER\_OF\_LOCAL\_LINES  
TYPES::BASIS\_TYPE, 1049  
NUMBER\_OF\_MATRICES  
TYPES::SOLVER\_MATRICES\_TYPE, 1323  
NUMBER\_OF\_MESHES  
TYPES::MESHS\_TYPE, 1272  
NUMBER\_OF\_NODE\_DOFS  
TYPES::FIELD\_DOF\_TO\_PARAM\_MAP\_-  
TYPE, 1201  
NUMBER\_OF\_NODE\_LINES  
TYPES::DOMAIN\_NODE\_TYPE, 1112  
NUMBER\_OF\_NODE\_PARAMETERS  
TYPES::FIELD\_PARAM\_TO\_DOF\_MAP\_-  
TYPE, 1215  
NUMBER\_OF\_NODES  
FIELD\_IO\_ROUTINES::FIELD\_IO\_-  
NODAL\_INFO\_SET, 848  
TYPES::BASIS\_TYPE, 1049  
TYPES::DOMAIN\_NODES\_TYPE, 1115  
TYPES::MESH\_NODES\_TYPE, 1263  
TYPES::NODES\_TYPE, 1276  
NUMBER\_OF\_NODES\_IN\_LOCAL\_LINE  
TYPES::BASIS\_TYPE, 1049  
NUMBER\_OF\_NODES\_XI  
TYPES::BASIS\_TYPE, 1049  
NUMBER\_OF\_PARAMETER\_SETS  
TYPES::FIELD\_PARAMETER\_SETS\_-  
TYPE, 1219  
NUMBER\_OF\_PARAMETERS  
TYPES::FIELD\_INTERPOLATION\_-  
PARAMETERS\_TYPE, 1211  
NUMBER\_OF\_PARTIAL\_DERIVATIVES  
TYPES::BASIS\_TYPE, 1050  
NUMBER\_OF\_POINT\_DOFS  
TYPES::FIELD\_DOF\_TO\_PARAM\_MAP\_-  
TYPE, 1201  
NUMBER\_OF\_POINT\_PARAMETERS  
TYPES::FIELD\_PARAM\_TO\_DOF\_MAP\_-  
TYPE, 1215  
NUMBER\_OF\_PROBLEM\_LINEARITIES  
PROBLEM\_ROUTINES\_LinearityTypes, 126  
NUMBER\_OF\_PROBLEM\_TIME\_TYPES  
PROBLEM\_ROUTINES\_-  
TimeDepedenceTypes, 128  
NUMBER\_OF\_PROBLEMS  
TYPES::PROBLEMS\_TYPE, 1293  
NUMBER\_OF\_RECEIVE\_GHOSTS  
TYPES::DOMAIN\_ADJACENT\_-  
DOMAIN\_TYPE, 1088  
NUMBER\_OF\_ROWS  
TYPES::ELEMENT\_MATRIX\_TYPE, 1123  
TYPES::ELEMENT\_VECTOR\_TYPE, 1124  
TYPES::EQUATIONS\_MAPPING\_TYPE,  
1150  
TYPES::EQUATIONS\_MATRICES\_TYPE,  
1160  
TYPES::SOLUTION\_MAPPING\_TYPE,  
1307  
TYPES::SOLVER\_MATRICES\_TYPE, 1323  
TYPES::SOLVER\_ROW\_TO\_-  
EQUATIONS\_SET\_MAP\_TYPE,  
1328  
NUMBER\_OF\_SCALING\_INDICES  
TYPES::FIELD\_SCALINGS\_TYPE, 1224  
NUMBER\_OF\_SCHEMES  
TYPES::QUADRATURE\_TYPE, 1298  
NUMBER\_OF\_SEND\_GHOSTS  
TYPES::DOMAIN\_ADJACENT\_-  
DOMAIN\_TYPE, 1088  
NUMBER\_OF\_SOLUTIONS  
TYPES::PROBLEM\_TYPE, 1291  
NUMBER\_OF\_SOLVER\_COLS  
TYPES::EQUATIONS\_COL\_TO\_SOLVER\_-  
COLS\_MAP\_TYPE, 1125  
TYPES::JACOBIAN\_COL\_TO\_SOLVER\_-  
COLS\_MAP\_TYPE, 1243  
NUMBER\_OF\_SOLVER\_MATRICES  
TYPES::EQUATIONS\_TO\_SOLVER\_-  
MATRIX\_MAPS\_EM\_TYPE, 1188  
TYPES::SOLUTION\_MAPPING\_TYPE,  
1307  
NUMBER\_OF\_SOLVER\_ROWS

TYPES::EQUATIONS\_ROW\_TO\_-  
     SOLVER\_ROWS\_MAP\_TYPE, 1168  
 NUMBER\_OF\_SUB\_BASES  
     TYPES::BASIS\_TYPE, 1050  
 NUMBER\_OF\_SUB\_REGIONS  
     TYPES::REGION\_TYPE, 1301  
 NUMBER\_OF\_SURROUNDING\_ELEMENTS  
     TYPES::DECOMPOSITION\_LINE\_TYPE,  
         1060  
     TYPES::DOMAIN\_NODE\_TYPE, 1112  
     TYPES::MESH\_NODE\_TYPE, 1262  
 NUMBER\_OF\_VARIABLES  
     TYPES::EQUATIONS\_TO\_SOLVER\_-  
         MATRIX\_MAPS\_SM\_TYPE, 1192  
     TYPES::FIELD\_TYPE, 1227  
 NUMBER\_OF\_VOLUMES  
     TYPES::FIELD\_GEOMETRIC\_-  
         PARAMETERS\_TYPE, 1204  
 NUMBER\_OF\_X\_DIMENSIONS  
     TYPES::FIELD\_INTERPOLATED\_POINT\_-  
         METRICS\_TYPE, 1206  
 NUMBER\_OF\_XI  
     TYPES::BASIS\_TYPE, 1050  
 NUMBER\_OF\_XI\_COORDINATES  
     TYPES::BASIS\_TYPE, 1050  
 NUMBER\_OF\_XI\_DIMENSIONS  
     TYPES::FIELD\_INTERPOLATED\_POINT\_-  
         METRICS\_TYPE, 1206  
 NUMBER\_PROCESSORS  
     COMP\_ENVIRONMENT::COMPUTATIONAL\_op\_eq\_CH\_VS  
         NODE\_TYPE, 824  
 NUMBER\_TO\_CHARACTER\_DP  
     STRINGS, 737  
     STRINGS::NUMBER\_TO\_CHARACTER,  
         1024  
 NUMBER\_TO\_CHARACTER\_INTG  
     STRINGS, 737  
     STRINGS::NUMBER\_TO\_CHARACTER,  
         1024  
 NUMBER\_TO\_CHARACTER\_LINTG  
     STRINGS, 738  
     STRINGS::NUMBER\_TO\_CHARACTER,  
         1024  
 NUMBER\_TO\_CHARACTER\_SP  
     STRINGS, 738  
     STRINGS::NUMBER\_TO\_CHARACTER,  
         1025  
 NUMBER\_TO\_VSTRING\_DP  
     STRINGS, 739  
     STRINGS::NUMBER\_TO\_VSTRING, 1026  
 NUMBER\_TO\_VSTRING\_INTG  
     STRINGS, 739  
     STRINGS::NUMBER\_TO\_VSTRING, 1026  
 NUMBER\_TO\_VSTRING\_LINTG

STRINGS, 740  
 STRINGS::NUMBER\_TO\_VSTRING, 1026  
 NUMBER\_TO\_VSTRING\_SP  
     STRINGS, 740  
     STRINGS::NUMBER\_TO\_VSTRING, 1027  
 OFFDIAGONAL\_NUMBER\_NON\_ZEROS  
     TYPES::DISTRIBUTED\_MATRIX\_-  
         PETSC\_TYPE, 1072  
 op\_assign\_CH\_VS  
     ISO\_VARYING\_STRING, 503  
     ISO\_VARYING\_STRING::assignment(=),  
         907  
 op\_assign\_VS\_CH  
     ISO\_VARYING\_STRING, 504  
     ISO\_VARYING\_STRING::assignment(=),  
         907  
 op\_concat\_CH\_VS  
     ISO\_VARYING\_STRING, 504  
     ISO\_VARYING\_STRING::assignment(=),  
         907  
 op\_concat\_VS\_CH  
     ISO\_VARYING\_STRING, 504  
     ISO\_VARYING\_STRING::assignment(=),  
         908  
 op\_concat\_VS\_VS  
     ISO\_VARYING\_STRING, 504  
     ISO\_VARYING\_STRING::assignment(=),  
         908  
 op\_eq\_CH\_VS  
     ISO\_VARYING\_STRING, 504  
     ISO\_VARYING\_STRING::assignment(=),  
         908  
 op\_eq\_VS\_CH  
     ISO\_VARYING\_STRING, 504  
     ISO\_VARYING\_STRING::assignment(=),  
         908  
 op\_eq\_VS\_VS  
     ISO\_VARYING\_STRING, 504  
     ISO\_VARYING\_STRING::assignment(=),  
         908  
 op\_ge\_CH\_VS  
     ISO\_VARYING\_STRING, 505  
     ISO\_VARYING\_STRING::assignment(=),  
         908  
 op\_ge\_VS\_CH  
     ISO\_VARYING\_STRING, 505  
     ISO\_VARYING\_STRING::assignment(=),  
         908  
 op\_ge\_VS\_VS  
     ISO\_VARYING\_STRING, 505  
     ISO\_VARYING\_STRING::assignment(=),  
         908  
 op\_gt\_CH\_VS

ISO\_VARYING\_STRING, 505  
ISO\_VARYING\_STRING::assignment(=),  
908  
op\_gt\_VS\_CH  
  ISO\_VARYING\_STRING, 505  
  ISO\_VARYING\_STRING::assignment(=),  
  909  
op\_gt\_VS\_VS  
  ISO\_VARYING\_STRING, 505  
  ISO\_VARYING\_STRING::assignment(=),  
  909  
op\_le\_CH\_VS  
  ISO\_VARYING\_STRING, 505  
  ISO\_VARYING\_STRING::assignment(=),  
  909  
op\_le\_VS\_CH  
  ISO\_VARYING\_STRING, 505  
  ISO\_VARYING\_STRING::assignment(=),  
  909  
op\_le\_VS\_VS  
  ISO\_VARYING\_STRING, 505  
  ISO\_VARYING\_STRING::assignment(=),  
  909  
op\_lt\_CH\_VS  
  ISO\_VARYING\_STRING, 506  
  ISO\_VARYING\_STRING::assignment(=),  
  909  
op\_lt\_VS\_CH  
  ISO\_VARYING\_STRING, 506  
  ISO\_VARYING\_STRING::assignment(=),  
  909  
op\_lt\_VS\_VS  
  ISO\_VARYING\_STRING, 506  
  ISO\_VARYING\_STRING::assignment(=),  
  909  
op\_ne\_CH\_VS  
  ISO\_VARYING\_STRING, 506  
  ISO\_VARYING\_STRING::assignment(=),  
  909  
op\_ne\_VS\_CH  
  ISO\_VARYING\_STRING, 506  
  ISO\_VARYING\_STRING::assignment(=),  
  910  
op\_ne\_VS\_VS  
  ISO\_VARYING\_STRING, 506  
  ISO\_VARYING\_STRING::assignment(=),  
  910  
OP\_STRING  
  BASE\_ROUTINES, 210  
OPEN\_BINARY\_FILE  
  BINARY\_FILE, 215  
OPEN\_CMISS\_BINARY\_FILE  
  BINARY\_FILE, 215  
OPEN\_COMFILE\_UNIT

BASE\_ROUTINES\_FileUnits, 24  
ORIENTATION  
  TYPES::COORDINATE\_SYSTEM\_TYPE,  
  1053  
ORIGIN  
  TYPES::COORDINATE\_SYSTEM\_TYPE,  
  1053  
  TYPES::GENERATED\_MESH\_-  
  REGULAR\_TYPE, 1239  
OS\_TYPE  
  BINARY\_FILE::BINARY\_FILE\_INFO\_-  
  TYPE, 781  
OUTPUT\_SET\_OFF  
  BASE\_ROUTINES, 205  
OUTPUT\_SET\_ON  
  BASE\_ROUTINES, 205  
OUTPUT\_TYPE  
  TYPES::EQUATIONS\_TYPE, 1196  
  TYPES::SOLVER\_TYPE, 1331  
PACKCHARACTERS  
  F90C::interface, 844  
PackCharacters  
  f90c\_c.c, 1412  
PARAM\_TO\_DOF\_MAP  
  TYPES::FIELD\_VARIABLE\_-  
  COMPONENT\_TYPE, 1230  
PARAMETER\_SETS  
  TYPES::FIELD\_PARAMETER\_SETS\_-  
  TYPE, 1219  
  TYPES::FIELD\_TYPE, 1227  
PARAMETERS  
  TYPES::FIELD\_INTERPOLATION\_-  
  PARAMETERS\_TYPE, 1211  
  TYPES::FIELD\_PARAMETER\_SET\_TYPE,  
  1217  
PARENT  
  TREES::TREE\_NODE\_TYPE, 1036  
PARENT\_BASIS  
  TYPES::BASIS\_TYPE, 1050  
PARENT\_REGION  
  TYPES::REGION\_TYPE, 1301  
PARMETIS\_PARTKWAY  
  CMISS\_PARMETIS, 237  
PARMETIS\_PARTMESHKWAY  
  CMISS\_PARMETIS, 237  
ParMETIS\_V3\_PartK  
  CMISS\_PARMETIS::interface, 798  
ParMETIS\_V3\_PartMeshKway  
  CMISS\_PARMETIS::interface, 798  
PARTIAL\_DERIVATIVE\_INDEX  
  TYPES::BASIS\_TYPE, 1050  
  TYPES::DOMAIN\_NODE\_TYPE, 1112  
  TYPES::MESH\_NODE\_TYPE, 1262

PARTIAL\_DERIVATIVE\_TYPE  
 TYPES::FIELD\_INTERPOLATED\_POINT\_TYPE, 1208

PC  
 TYPES::LINEAR\_ITERATIVE\_SOLVER\_TYPE, 1248

PCSetType  
 CMISS\_PETSC::interface, 805

PETSC  
 TYPES::DISTRIBUTED\_MATRIX\_TYPE, 1074  
 TYPES::DISTRIBUTED\_VECTOR\_TYPE, 1086

PETSC\_ERRORHANDLING\_SET\_OFF  
 CMISS\_PETSC, 246

PETSC\_ERRORHANDLING\_SET\_ON  
 CMISS\_PETSC, 246

PETSC\_FINALIZE  
 CMISS\_PETSC, 246

PETSC\_HANDLE\_ERROR  
 CMISS\_PETSC, 291

PETSC\_INITIALIZE  
 CMISS\_PETSC, 247

PETSC\_ISCOLORINGDESTROY  
 CMISS\_PETSC, 247

PETSC\_ISCOLORINGFINALISE  
 CMISS\_PETSC, 248

PETSC\_ISCOLORINGINITIALISE  
 CMISS\_PETSC, 248

PETSC\_ISDESTROY  
 CMISS\_PETSC, 248

PETSC\_ISFINALISE  
 CMISS\_PETSC, 249

PETSC\_ISINITIALISE  
 CMISS\_PETSC, 249

PETSC\_ISLOCALTOGLOBALMAPPINGAPPLY  
 CMISS\_PETSC, 249

PETSC\_ISLOCALTOGLOBALMAPPINGAPPLYIS  
 CMISS\_PETSC, 250

PETSC\_ISLOCALTOGLOBALMAPPINGCREATE  
 CMISS\_PETSC, 250

PETSC\_ISLOCALTOGLOBALMAPPINGDESTROY  
 CMISS\_PETSC, 250

PETSC\_ISLOCALTOGLOBALMAPPINGFINALISE  
 CMISS\_PETSC, 251

PETSC\_ISLOCALTOGLOBALMAPPINGINITIALISE  
 CMISS\_PETSC, 251

PETSC\_KSPCREATE  
 CMISS\_PETSC, 251

PETSC\_KSPDESTROY  
 CMISS\_PETSC, 252

PETSC\_KSPFINALISE  
 CMISS\_PETSC, 252

PETSC\_KSPGETCONVERGEDREASON  
 CMISS\_PETSC, 253

PETSC\_KSPGETITERATIONNUMBER  
 CMISS\_PETSC, 253

PETSC\_KSPGETPC  
 CMISS\_PETSC, 253

PETSC\_KSPGETRESIDUALNORM  
 CMISS\_PETSC, 254

PETSC\_KSPINITIALISE  
 CMISS\_PETSC, 254

PETSC\_KSPSETFROMOPTIONS  
 CMISS\_PETSC, 255

PETSC\_KSPSETOPERATORS  
 CMISS\_PETSC, 255

PETSC\_KSPSETTOLERANCES  
 CMISS\_PETSC, 256

PETSC\_KSPSETTYPE  
 CMISS\_PETSC, 256

PETSC\_KSPSETUP  
 CMISS\_PETSC, 256

PETSC\_KSPSOLVE  
 CMISS\_PETSC, 257

PETSC\_LOGPRINTSUMMARY  
 CMISS\_PETSC, 257

PETSC\_MATASSEMBLYBEGIN  
 CMISS\_PETSC, 258

PETSC\_MATASSEMBLYEND  
 CMISS\_PETSC, 258

PETSC\_MATCREATE  
 CMISS\_PETSC, 258

PETSC\_MATCREATEMPIAIJ  
 CMISS\_PETSC, 259

PETSC\_MATCREATEMPIDENSE  
 CMISS\_PETSC, 259

PETSC\_MATCREATESEQAIJ  
 CMISS\_PETSC, 260

PETSC\_MATCREATESEQDENSE  
 CMISS\_PETSC, 260

PETSC\_MATDESTROY  
 CMISS\_PETSC, 261

PETSC\_MATFDCOLORINGCREATE  
 CMISS\_PETSC, 261

PETSC\_MATFDCOLORINGDESTROY  
 CMISS\_PETSC, 262

PETSC\_MATFDCOLORINGFINALISE  
 CMISS\_PETSC, 262

PETSC\_MATFDCOLORINGINITIALISE  
 CMISS\_PETSC, 262

PETSC\_MATFDCOLORINGSETFROMOPTIONS  
 CMISS\_PETSC, 263

PETSC\_MATFINALISE  
 CMISS\_PETSC, 263

PETSC\_MATGETARRAY  
 CMISS\_PETSC, 263

PETSC\_MATGETCOLORING  
 CMISS\_PETSC, 263

CMISS\_PETSC, 264  
PETSC\_MATGETOWNERSHIPRANGE  
    CMISS\_PETSC, 264  
PETSC\_MATGETVALUES  
    CMISS\_PETSC, 265  
PETSC\_MATINITIALISE  
    CMISS\_PETSC, 265  
PETSC\_MATRESTOREARRAY  
    CMISS\_PETSC, 265  
PETSC\_MATSETLOCALTOGLOBALMAPPING  
    CMISS\_PETSC, 266  
PETSC\_MATSETOPTION  
    CMISS\_PETSC, 266  
PETSC\_MATSETSIZES  
    CMISS\_PETSC, 267  
PETSC\_MATSETVALUE  
    CMISS\_PETSC, 267  
PETSC\_MATSETVALUELOCAL  
    CMISS\_PETSC, 267  
PETSC\_MATSETVALUES  
    CMISS\_PETSC, 268  
PETSC\_MATSETVALUESLOCAL  
    CMISS\_PETSC, 268  
PETSC\_MATVIEW  
    CMISS\_PETSC, 269  
PETSC\_MATZEROENTRIES  
    CMISS\_PETSC, 269  
PETSC\_PCFINALISE  
    CMISS\_PETSC, 270  
PETSC\_PCINITIALISE  
    CMISS\_PETSC, 270  
PETSC\_PCSETTYPE  
    CMISS\_PETSC, 270  
PETSC\_SNES\_LINESEARCH\_CUBIC  
    CMISS\_PETSC, 291  
PETSC\_SNES\_LINESEARCH\_NO  
    CMISS\_PETSC, 291  
PETSC\_SNES\_LINESEARCH\_NONORMS  
    CMISS\_PETSC, 291  
PETSC\_SNES\_LINESEARCH\_QUADRATIC  
    CMISS\_PETSC, 291  
PETSC\_SNESCREATE  
    CMISS\_PETSC, 271  
PETSC\_SNESDESTROY  
    CMISS\_PETSC, 271  
PETSC\_SNESFINALISE  
    CMISS\_PETSC, 272  
PETSC\_SNESGETCONVERGEDREASON  
    CMISS\_PETSC, 272  
PETSC\_SNESGETFUNCTIONNORM  
    CMISS\_PETSC, 272  
PETSC\_SNESGETITERATIONNUMBER  
    CMISS\_PETSC, 273  
PETSC\_SNESINITIALISE  
    CMISS\_PETSC, 273  
PETSC\_SNESLINESEARCHSET  
    CMISS\_PETSC, 274  
PETSC\_SNESLINESEARCHSETPARAMS  
    CMISS\_PETSC, 274  
PETSC\_SNESSETFROMOPTIONS  
    CMISS\_PETSC, 275  
PETSC\_SNESSETFUNCTION  
    CMISS\_PETSC, 275  
PETSC\_SNESSETJACOBIAN\_-  
    MATFDCOLORING  
    CMISS\_PETSC, 276  
    CMISS\_PETSC::PETSC\_-  
    SNESSETJACOBIAN, 811  
PETSC\_SNESSETJACOBIAN\_SOLVER  
    CMISS\_PETSC, 276  
    CMISS\_PETSC::PETSC\_-  
    SNESSETJACOBIAN, 811  
PETSC\_SNESSETTOLERANCES  
    CMISS\_PETSC, 276  
PETSC\_SNESSETTRUSTREGIONTOLERANCE  
    CMISS\_PETSC, 277  
PETSC\_SNESSETTYPE  
    CMISS\_PETSC, 277  
PETSC\_SNESSOLVE  
    CMISS\_PETSC, 278  
PETSC\_VECASSEMBLYBEGIN  
    CMISS\_PETSC, 278  
PETSC\_VECASSEMBLYEND  
    CMISS\_PETSC, 279  
PETSC\_VECCREATE  
    CMISS\_PETSC, 279  
PETSC\_VECCREATEGHOST  
    CMISS\_PETSC, 279  
PETSC\_VECCREATEGHOSTWITHARRAY  
    CMISS\_PETSC, 280  
PETSC\_VECCREATEMPI  
    CMISS\_PETSC, 280  
PETSC\_VECCREATEMPIWITHARRAY  
    CMISS\_PETSC, 281  
PETSC\_VECCREATESEQ  
    CMISS\_PETSC, 281  
PETSC\_VECCREATESEQWITHARRAY  
    CMISS\_PETSC, 282  
PETSC\_VECDESTROY  
    CMISS\_PETSC, 282  
PETSC\_VECDUPLICATE  
    CMISS\_PETSC, 282  
PETSC\_VECFINALISE  
    CMISS\_PETSC, 283  
PETSC\_VECGETARRAY  
    CMISS\_PETSC, 283  
PETSC\_VECGETARRAYF90  
    CMISS\_PETSC, 283

PETSC\_VECGETLOCALSIZE  
     CMISS\_PETSC, 284  
 PETSC\_VECGETOWNERSHIPRANGE  
     CMISS\_PETSC, 284  
 PETSC\_VECGETSIZE  
     CMISS\_PETSC, 285  
 PETSC\_VECGETVALUES  
     CMISS\_PETSC, 285  
 PETSC\_VECGHOSTGETLOCALFORM  
     CMISS\_PETSC, 285  
 PETSC\_VECGHOSTRESTORELOCALFORM  
     CMISS\_PETSC, 286  
 PETSC\_VECGHOSTUPDATEBEGIN  
     CMISS\_PETSC, 286  
 PETSC\_VECGHOSTUPDATEEND  
     CMISS\_PETSC, 287  
 PETSC\_VECINITIALISE  
     CMISS\_PETSC, 287  
 PETSC\_VECRESTOREARRAY  
     CMISS\_PETSC, 287  
 PETSC\_VECRESTOREARRAYF90  
     CMISS\_PETSC, 288  
 PETSC\_VECSET  
     CMISS\_PETSC, 288  
 PETSC\_VECSETFROMOPTIONS  
     CMISS\_PETSC, 288  
 PETSC\_VECSETLOCALTOGLOBALMAPPING  
     CMISS\_PETSC, 289  
 PETSC\_VECSETSIZES  
     CMISS\_PETSC, 289  
 PETSC\_VECSETVALUES  
     CMISS\_PETSC, 290  
 PETSC\_VECSETVALUESLOCAL  
     CMISS\_PETSC, 290  
 PETSC\_VECVIEW  
     CMISS\_PETSC, 291  
 PetscFinalize  
     CMISS\_PETSC::interface, 805  
 PetscInitialize  
     CMISS\_PETSC::interface, 805  
 PetscLogPrintSummary  
     CMISS\_PETSC::interface, 805  
 POINT\_DOF2PARAM\_MAP  
     TYPES::FIELD\_DOF\_TO\_PARAM\_MAP\_-  
         TYPE, 1201  
 POINT\_PARAM2DOF\_MAP  
     TYPES::FIELD\_PARAM\_TO\_DOF\_MAP\_-  
         TYPE, 1215  
 PREVIOUS\_ROUTINE  
     BASE\_ROUTINES::ROUTINE\_STACK\_-  
         ITEM\_TYPE, 777  
 PROBLEM  
     TYPES::PROBLEM\_CONTROL\_TYPE,  
         1285  
             TYPES::SOLUTION\_TYPE, 1311  
 PROBLEM\_ADVECTION\_DIFFUSION\_-  
     EQUATION\_TYPE  
         PROBLEM\_CONSTANTS, 647  
 PROBLEM\_BIHAMONIC\_EQUATION\_TYPE  
         PROBLEM\_CONSTANTS, 647  
 PROBLEM\_CLASSICAL\_FIELD\_CLASS  
         PROBLEM\_CONSTANTS, 648  
 PROBLEM\_CONSTANTS, 646  
     PROBLEM\_ADVECTION\_DIFFUSION\_-  
         EQUATION\_TYPE, 647  
     PROBLEM\_BIHAMONIC\_EQUATION\_-  
         TYPE, 647  
     PROBLEM\_CLASSICAL\_FIELD\_CLASS,  
         648  
     PROBLEM\_DIFFUSION\_EQUATION\_-  
         TYPE, 648  
     PROBLEM\_ELASTICITY\_CLASS, 648  
     PROBLEM\_ELECTROMAGNETICS\_-  
         CLASS, 648  
     PROBLEM\_ELECTROSTATIC\_TYPE, 648  
     PROBLEMFINITE\_ELASTICITY\_TYPE,  
         648  
     PROBLEM\_FITTING\_CLASS, 649  
     PROBLEM\_FLUID\_MECHANICS\_CLASS,  
         649  
     PROBLEM\_GENERALISED\_LAPLACE\_-  
         SUBTYPE, 649  
     PROBLEM\_HELMHOLTZ\_EQUATION\_-  
         TYPE, 649  
     PROBLEM\_LAPLACE\_EQUATION\_TYPE,  
         649  
     PROBLEM\_LINEAR\_ELASTIC\_MODAL\_-  
         TYPE, 649  
     PROBLEM\_LINEAR\_ELASTICITY\_TYPE,  
         650  
     PROBLEM\_MAGNETOSTATIC\_TYPE, 650  
     PROBLEM\_MAXWELLS\_EQUATIONS\_-  
         TYPE, 650  
     PROBLEM\_MODAL\_CLASS, 650  
     PROBLEM\_NAVIER\_STOKES\_FLUID\_-  
         TYPE, 650  
     PROBLEM\_NO\_CLASS, 650  
     PROBLEM\_NO\_SUBTYPE, 650  
     PROBLEM\_NO\_TYPE, 650  
     PROBLEM\_OPTIMISATION\_CLASS, 651  
     PROBLEM\_POISSON\_EQUATION\_TYPE,  
         651  
     PROBLEM\_REACTION\_DIFFUSION\_-  
         EQUATION\_TYPE, 651  
     PROBLEM\_STANDARD\_LAPLACE\_-  
         SUBTYPE, 651  
     PROBLEM\_STOKES\_FLUID\_TYPE, 651  
     PROBLEM\_WAVE\_EQUATION\_TYPE, 651

PROBLEM\_CONSTANTS::SetupActionTypes, 117  
PROBLEM\_CONSTANTS::SetupTypes, 115  
PROBLEM\_CONSTANTS::SolutionLinearityTypes, 121  
PROBLEM\_CONSTANTS::SolutionOutputTypes, 119  
PROBLEM\_CONSTANTS::SolverOutputTypes, 123  
PROBLEM\_CONSTANTS::SolverSparsityTypes, 125  
PROBLEM\_CONSTANTS\_SetupActionTypes  
    PROBLEM\_SETUP\_DO\_ACTION, 117  
    PROBLEM\_SETUP\_FINISH\_ACTION, 117  
    PROBLEM\_SETUP\_START\_ACTION, 118  
PROBLEM\_CONSTANTS\_SetupTypes  
    PROBLEM\_SETUP\_CONTROL\_TYPE, 115  
    PROBLEM\_SETUP\_INITIAL\_TYPE, 115  
    PROBLEM\_SETUP SOLUTION\_TYPE, 116  
    PROBLEM\_SETUP\_SOLVER\_TYPE, 116  
PROBLEM\_CONSTANTS\_-  
    SolutionLinearityTypes  
    PROBLEM SOLUTION\_LINEAR, 121  
    PROBLEM SOLUTION\_NONLINEAR, 121  
PROBLEM\_CONSTANTS\_SolutionOutputTypes  
    PROBLEM SOLUTION MATRIX\_-  
        OUTPUT, 119  
    PROBLEM SOLUTION\_NO\_OUTPUT, 119  
    PROBLEM SOLUTION\_TIMING\_-  
        OUTPUT, 119  
PROBLEM\_CONSTANTS\_SolverOutputTypes  
    PROBLEM SOLVER\_NO\_OUTPUT, 123  
    PROBLEM SOLVER\_SOLVER\_OUTPUT,  
        123  
    PROBLEM SOLVER\_TIMING\_OUTPUT,  
        123  
PROBLEM\_CONSTANTS\_SolverSparsityTypes  
    PROBLEM SOLVER\_FULL\_MATRICES,  
        125  
    PROBLEM SOLVER\_SPARSE\_-  
        MATRICES, 125  
PROBLEM\_CONTROL\_CREATE\_FINISH  
    PROBLEM\_ROUTINES, 655  
PROBLEM\_CONTROL\_CREATE\_START  
    PROBLEM\_ROUTINES, 655  
PROBLEM\_CONTROL\_DESTROY  
    PROBLEM\_ROUTINES, 655  
PROBLEM\_CONTROL\_FINALISE  
    PROBLEM\_ROUTINES, 656  
PROBLEM\_CONTROL\_INITIALISE  
    PROBLEM\_ROUTINES, 656  
PROBLEM\_CREATE\_FINISH  
    PROBLEM\_ROUTINES, 657  
PROBLEM\_CREATE\_START  
    PROBLEM\_ROUTINES, 657  
PROBLEM\_ROUTINES, 657  
PROBLEM\_DESTROY\_NUMBER  
    PROBLEM\_ROUTINES, 658  
PROBLEM\_ROUTINES::PROBLEM\_-  
    DESTROY, 1007  
PROBLEM\_DESTROY\_PTR  
    PROBLEM\_ROUTINES, 658  
PROBLEM\_ROUTINES::PROBLEM\_-  
    DESTROY, 1007  
PROBLEM\_DIFFUSION\_EQUATION\_TYPE  
    PROBLEM\_CONSTANTS, 648  
PROBLEM\_DYNAMIC  
    PROBLEM\_ROUTINES\_-  
        TimeDepedenceTypes, 128  
PROBLEM\_ELASTICITY\_CLASS  
    PROBLEM\_CONSTANTS, 648  
PROBLEM\_ELECTROMAGNETICS\_CLASS  
    PROBLEM\_CONSTANTS, 648  
PROBLEM\_ELECTROSTATIC\_TYPE  
    PROBLEM\_CONSTANTS, 648  
PROBLEM\_FINALISE  
    PROBLEM\_ROUTINES, 658  
PROBLEM\_FINISHED  
    TYPES::PROBLEM\_TYPE, 1291  
PROBLEMFINITE\_ELASTICITY\_TYPE  
    PROBLEM\_CONSTANTS, 648  
PROBLEM\_FITTING\_CLASS  
    PROBLEM\_CONSTANTS, 649  
PROBLEM\_FLUID\_MECHANICS\_CLASS  
    PROBLEM\_CONSTANTS, 649  
PROBLEM\_GENERALISED\_LAPLACE\_-  
    SUBTYPE  
    PROBLEM\_CONSTANTS, 649  
PROBLEM\_HELMHOLTZ\_EQUATION\_TYPE  
    PROBLEM\_CONSTANTS, 649  
PROBLEM\_INITIALISE  
    PROBLEM\_ROUTINES, 659  
PROBLEM\_LAPLACE\_EQUATION\_TYPE  
    PROBLEM\_CONSTANTS, 649  
PROBLEM\_LINEAR  
    PROBLEM\_ROUTINES\_LinearityTypes, 126  
PROBLEM\_LINEAR\_ELASTIC\_MODAL\_TYPE  
    PROBLEM\_CONSTANTS, 649  
PROBLEM\_LINEAR\_ELASTICITY\_TYPE  
    PROBLEM\_CONSTANTS, 650  
PROBLEM\_MAGNETOSTATIC\_TYPE  
    PROBLEM\_CONSTANTS, 650  
PROBLEM\_MAXWELLS\_EQUATIONS\_TYPE  
    PROBLEM\_CONSTANTS, 650  
PROBLEM\_MODAL\_CLASS  
    PROBLEM\_CONSTANTS, 650  
PROBLEM\_NAVIER\_STOKES\_FLUID\_TYPE  
    PROBLEM\_CONSTANTS, 650  
PROBLEM\_NO\_CLASS

PROBLEM\_CONSTANTS, 650  
 PROBLEM\_NO\_SUBTYPE  
     PROBLEM\_CONSTANTS, 650  
 PROBLEM\_NO\_TYPE  
     PROBLEM\_CONSTANTS, 650  
 PROBLEM\_NONLINEAR  
     PROBLEM\_ROUTINES\_LinearityTypes, 126  
 PROBLEM\_NONLINEAR\_BCS  
     PROBLEM\_ROUTINES\_LinearityTypes, 127  
 PROBLEM\_OPTIMISATION\_CLASS  
     PROBLEM\_CONSTANTS, 651  
 PROBLEM\_POISSON\_EQUATION\_TYPE  
     PROBLEM\_CONSTANTS, 651  
 PROBLEM\_QUASISTATIC  
     PROBLEM\_ROUTINES\_-  
         TimeDepedenceTypes, 128  
 PROBLEM\_REACTION\_DIFFUSION\_-  
     EQUATION\_TYPE  
     PROBLEM\_CONSTANTS, 651  
 PROBLEM\_ROUTINES, 652  
     PROBLEM\_CONTROL\_CREATE\_FINISH,  
         655  
     PROBLEM\_CONTROL\_CREATE\_START,  
         655  
     PROBLEM\_CONTROL\_DESTROY, 655  
     PROBLEM\_CONTROL\_FINALISE, 656  
     PROBLEM\_CONTROL\_INITIALISE, 656  
     PROBLEM\_CREATE\_FINISH, 657  
     PROBLEM\_CREATE\_START, 657  
     PROBLEM\_DESTROY\_NUMBER, 658  
     PROBLEM\_DESTROY\_PTR, 658  
     PROBLEM\_FINALISE, 658  
     PROBLEM\_INITIALISE, 659  
     PROBLEM\_SETUP, 659  
     PROBLEM\_SOLUTION\_EQUATIONS\_-  
         SET\_ADD, 660  
     PROBLEM\_SOLUTION\_FINALISE, 660  
     PROBLEM\_SOLUTION\_INITIALISE, 661  
     PROBLEM\_SOLUTION\_JACOBIAN\_-  
         EVALUATE, 661  
     PROBLEM\_SOLUTION\_RESIDUAL\_-  
         EVALUATE, 662  
     PROBLEM\_SOLUTION\_SOLVE, 662  
     PROBLEM\_SOLUTIONS\_CREATE\_-  
         FINISH, 663  
     PROBLEM\_SOLUTIONS\_CREATE\_-  
         START, 663  
     PROBLEM\_SOLUTIONS\_FINALISE, 663  
     PROBLEM\_SOLUTIONS\_INITIALISE, 664  
     PROBLEM\_SOLVE, 664  
     PROBLEM\_SOLVER\_CREATE\_FINISH,  
         665  
     PROBLEM\_SOLVER\_CREATE\_START, 665  
     PROBLEM\_SOLVER\_DESTROY, 665  
     PROBLEM\_SOLVER\_GET, 666  
     PROBLEM\_SPECIFICATION\_GET\_-  
         NUMBER, 666  
     PROBLEM\_SPECIFICATION\_GET\_PTR,  
         667  
     PROBLEM\_SPECIFICATION\_SET\_-  
         NUMBER, 667  
     PROBLEM\_SPECIFICATION\_SET\_PTR,  
         668  
     PROBLEM\_USER\_NUMBER\_FIND, 668  
 PROBLEMS, 670  
 PROBLEMS\_FINALISE, 669  
 PROBLEMS\_INITIALISE, 669  
 problem\_routines.f90  
     PROBLEM\_SOLUTION\_JACOBIAN\_-  
         EVALUATE\_PETSC, 1508  
     PROBLEM\_SOLUTION\_RESIDUAL\_-  
         EVALUATE\_PETSC, 1509  
 PROBLEM\_ROUTINES::LinearityTypes, 126  
 PROBLEM\_ROUTINES::PROBLEM\_DESTROY,  
     1007  
     PROBLEM\_DESTROY\_NUMBER, 1007  
     PROBLEM\_DESTROY\_PTR, 1007  
 PROBLEM\_ROUTINES::PROBLEM\_-  
     SPECIFICATION\_GET, 1008  
     PROBLEM\_SPECIFICATION\_GET\_-  
         NUMBER, 1008  
     PROBLEM\_SPECIFICATION\_GET\_PTR,  
         1008  
 PROBLEM\_ROUTINES::PROBLEM\_-  
     SPECIFICATION\_SET, 1010  
     PROBLEM\_SPECIFICATION\_SET\_-  
         NUMBER, 1010  
     PROBLEM\_SPECIFICATION\_SET\_PTR,  
         1010  
 PROBLEM\_ROUTINES::TimeDepedenceTypes,  
     128  
 PROBLEM\_ROUTINES\_LinearityTypes  
     NUMBER\_OF\_PROBLEM\_LINEARITIES,  
         126  
     PROBLEM\_LINEAR, 126  
     PROBLEM\_NONLINEAR, 126  
     PROBLEM\_NONLINEAR\_BCS, 127  
 PROBLEM\_ROUTINES\_TimeDepedenceTypes  
     NUMBER\_OF\_PROBLEM\_TIME\_TYPES,  
         128  
     PROBLEM\_DYNAMIC, 128  
     PROBLEM\_QUASISTATIC, 128  
     PROBLEM\_STATIC, 129  
 PROBLEM\_SETUP  
     PROBLEM\_ROUTINES, 659  
 PROBLEM\_SETUP\_CONTROL\_TYPE  
     PROBLEM\_CONSTANTS\_SetupTypes, 115  
 PROBLEM\_SETUP\_DO\_ACTION

PROBLEM\_CONSTANTS\_-  
  SetupActionTypes, 117  
PROBLEM\_SETUP\_FINISH\_ACTION  
  PROBLEM\_CONSTANTS\_-  
    SetupActionTypes, 117  
PROBLEM\_SETUP\_INITIAL\_TYPE  
  PROBLEM\_CONSTANTS\_SetupTypes, 115  
PROBLEM\_SETUP\_SOLUTION\_TYPE  
  PROBLEM\_CONSTANTS\_SetupTypes, 116  
PROBLEM\_SETUP\_SOLVER\_TYPE  
  PROBLEM\_CONSTANTS\_SetupTypes, 116  
PROBLEM\_SETUP\_START\_ACTION  
  PROBLEM\_CONSTANTS\_-  
    SetupActionTypes, 118  
PROBLEM SOLUTION EQUATIONS\_SET\_-  
  ADD  
    PROBLEM\_ROUTINES, 660  
PROBLEM SOLUTION FINALISE  
  PROBLEM\_ROUTINES, 660  
PROBLEM SOLUTION INITIALISE  
  PROBLEM\_ROUTINES, 661  
PROBLEM SOLUTION JACOBIAN\_-  
  EVALUATE  
    PROBLEM\_ROUTINES, 661  
PROBLEM SOLUTION JACOBIAN\_-  
  EVALUATE\_PETSC  
    problem\_routines.f90, 1508  
PROBLEM SOLUTION LINEAR  
  PROBLEM\_CONSTANTS\_-  
    SolutionLinearityTypes, 121  
PROBLEM SOLUTION MATRIX\_OUTPUT  
  PROBLEM\_CONSTANTS\_-  
    SolutionOutputTypes, 119  
PROBLEM SOLUTION NO\_OUTPUT  
  PROBLEM\_CONSTANTS\_-  
    SolutionOutputTypes, 119  
PROBLEM SOLUTION NONLINEAR  
  PROBLEM\_CONSTANTS\_-  
    SolutionLinearityTypes, 121  
PROBLEM SOLUTION RESIDUAL\_-  
  EVALUATE  
    PROBLEM\_ROUTINES, 662  
PROBLEM SOLUTION RESIDUAL\_-  
  EVALUATE\_PETSC  
    problem\_routines.f90, 1509  
PROBLEM SOLUTION SOLVE  
  PROBLEM\_ROUTINES, 662  
PROBLEM SOLUTION TIMING\_OUTPUT  
  PROBLEM\_CONSTANTS\_-  
    SolutionOutputTypes, 119  
PROBLEM SOLUTIONS CREATE FINISH  
  PROBLEM\_ROUTINES, 663  
PROBLEM SOLUTIONS CREATE START  
  PROBLEM\_ROUTINES, 663  
PROBLEM SOLUTIONS\_FINALISE  
  PROBLEM\_ROUTINES, 663  
PROBLEM SOLUTIONS\_INITIALISE  
  PROBLEM\_ROUTINES, 664  
PROBLEM SOLVE  
  PROBLEM\_ROUTINES, 664  
PROBLEM SOLVER\_CREATE\_FINISH  
  PROBLEM\_ROUTINES, 665  
PROBLEM SOLVER\_CREATE\_START  
  PROBLEM\_ROUTINES, 665  
PROBLEM SOLVER\_DESTROY  
  PROBLEM\_ROUTINES, 665  
PROBLEM SOLVER\_FULL\_MATRICES  
  PROBLEM\_CONSTANTS\_-  
    SolverSparsityTypes, 125  
PROBLEM SOLVER\_GET  
  PROBLEM\_ROUTINES, 666  
PROBLEM SOLVER\_NO\_OUTPUT  
  PROBLEM\_CONSTANTS\_-  
    SolverOutputTypes, 123  
PROBLEM SOLVER\_SOLVER\_OUTPUT  
  PROBLEM\_CONSTANTS\_-  
    SolverOutputTypes, 123  
PROBLEM SOLVER\_SPARSE\_MATRICES  
  PROBLEM\_CONSTANTS\_-  
    SolverSparsityTypes, 125  
PROBLEM SOLVER\_TIMING\_OUTPUT  
  PROBLEM\_CONSTANTS\_-  
    SolverOutputTypes, 123  
PROBLEM SPECIFICATION GET NUMBER  
  PROBLEM\_ROUTINES, 666  
  PROBLEM\_ROUTINES::PROBLEM\_-  
    SPECIFICATION\_GET, 1008  
PROBLEM SPECIFICATION GET PTR  
  PROBLEM\_ROUTINES, 667  
  PROBLEM\_ROUTINES::PROBLEM\_-  
    SPECIFICATION\_GET, 1008  
PROBLEM SPECIFICATION SET NUMBER  
  PROBLEM\_ROUTINES, 667  
  PROBLEM\_ROUTINES::PROBLEM\_-  
    SPECIFICATION\_SET, 1010  
PROBLEM SPECIFICATION SET PTR  
  PROBLEM\_ROUTINES, 668  
  PROBLEM\_ROUTINES::PROBLEM\_-  
    SPECIFICATION\_SET, 1010  
PROBLEM STANDARD LAPLACE SUBTYPE  
  PROBLEM\_CONSTANTS, 651  
PROBLEM STATIC  
  PROBLEM\_ROUTINES\_-  
    TimeDepedenceTypes, 129  
PROBLEM STOKES FLUID TYPE  
  PROBLEM\_CONSTANTS, 651  
PROBLEM USER NUMBER FIND  
  PROBLEM\_ROUTINES, 668

PROBLEM\_WAVE\_EQUATION\_TYPE  
   PROBLEM\_CONSTANTS, 651  
 PROBLEMS  
   PROBLEM\_ROUTINES, 670  
   TYPES::PROBLEM\_TYPE, 1291  
   TYPES::PROBLEMS\_TYPE, 1293  
 PROBLEMS\_FINALISE  
   PROBLEM\_ROUTINES, 669  
 PROBLEMS\_INITIALISE  
   PROBLEM\_ROUTINES, 669  
 PTR  
   COORDINATE\_-  
     ROUTINES::COORDINATE\_-  
       SYSTEM\_PTR\_TYPE, 837  
   FIELD\_IO\_ROUTINES::FIELD\_-  
     VARIABLE\_COMPONENT\_PTR\_-  
       TYPE, 850  
   FIELD\_IO\_ROUTINES::MESH\_-  
     ELEMENTS\_TYPE\_PTR\_TYPE,  
       851  
   KINDS\_IntegerKinds, 92  
   LISTS::LIST\_PTR\_TYPE, 945  
   TYPES::BASIS\_PTR\_TYPE, 1041  
   TYPES::DECOMPOSITION\_PTR\_TYPE,  
     1062  
   TYPES::DOMAIN\_FACE\_PTR\_TYPE, 1095  
   TYPES::DOMAIN\_LINE\_PTR\_TYPE, 1101  
   TYPES::DOMAIN\_PTR\_TYPE, 1116  
   TYPES::EQUATIONS\_MATRIX\_PTR\_-  
     TYPE, 1162  
   TYPES::EQUATIONS\_SET\_PTR\_TYPE,  
     1175  
   TYPES::EQUATIONS\_TO\_SOLVER\_-  
     MAPS\_PTR\_TYPE, 1185  
   TYPES::FIELD\_PARAMETER\_SET\_PTR\_-  
     TYPE, 1216  
   TYPES::FIELD\_PTR\_TYPE, 1221  
   TYPES::FIELD\_VARIABLE\_PTR\_TYPE,  
     1232  
   TYPES::GENERATED\_MESH\_PTR\_TYPE,  
     1237  
   TYPES::MESH\_PTR\_TYPE, 1264  
   TYPES::MESH\_TOPOLOGY\_PTR\_TYPE,  
     1265  
   TYPES::PROBLEM\_PTR\_TYPE, 1288  
   TYPES::QUADRATURE\_SCHEME\_PTR\_-  
     TYPE, 1294  
   TYPES::REGION\_PTR\_TYPE, 1299  
   TYPES::SOLUTION\_PTR\_TYPE, 1309  
   TYPES::SOLVER\_MATRIX\_PTR\_TYPE,  
     1325  
 put\_CH  
   ISO\_VARYING\_STRING, 506  
   ISO\_VARYING\_STRING::put, 925  
 put\_line\_CH  
   ISO\_VARYING\_STRING, 506  
   ISO\_VARYING\_STRING::put\_line, 926  
 put\_line\_unit\_CH  
   ISO\_VARYING\_STRING, 506  
   ISO\_VARYING\_STRING::put\_line, 926  
 put\_line\_unit\_VS  
   ISO\_VARYING\_STRING, 507  
   ISO\_VARYING\_STRING::put\_line, 926  
 put\_line\_VS  
   ISO\_VARYING\_STRING, 507  
   ISO\_VARYING\_STRING::put\_line, 926  
 put\_unit\_CH  
   ISO\_VARYING\_STRING, 507  
   ISO\_VARYING\_STRING::put, 925  
 put\_unit\_VS  
   ISO\_VARYING\_STRING, 507  
   ISO\_VARYING\_STRING::put, 925  
 put\_VS  
   ISO\_VARYING\_STRING, 507  
   ISO\_VARYING\_STRING::put, 925  
 QP  
   KINDS\_RealKinds, 95  
 QUADRATURE  
   TYPES::BASIS\_TYPE, 1050  
   TYPES::QUADRATURE\_SCHEME\_TYPE,  
     1296  
 QUADRATURE\_SCHEME\_MAP  
   TYPES::QUADRATURE\_TYPE, 1298  
 QUADRUPLECOMPLEXTYPE  
   binary\_file\_c.c, 1358  
 QUADRUPLETYP  
   binary\_file\_c.c, 1358  
 RADIAL\_INTERPOLATION\_TYPE  
   TYPES::COORDINATE\_SYSTEM\_TYPE,  
     1053  
 RANK  
   COMP\_ENVIRONMENT::COMPUTATIONAL\_-  
     NODE\_TYPE, 824  
 READ\_BINARY\_FILE\_CHARACTER  
   BINARY\_FILE, 216  
   BINARY\_FILE::READ\_BINARY\_FILE, 787  
 READ\_BINARY\_FILE\_DP  
   BINARY\_FILE, 216  
   BINARY\_FILE::READ\_BINARY\_FILE, 787  
 READ\_BINARY\_FILE\_DP1  
   BINARY\_FILE, 216  
   BINARY\_FILE::READ\_BINARY\_FILE, 787  
 READ\_BINARY\_FILE\_DPC  
   BINARY\_FILE, 216  
   BINARY\_FILE::READ\_BINARY\_FILE, 787  
 READ\_BINARY\_FILE\_DPC1

BINARY\_FILE, 217  
BINARY\_FILE::READ\_BINARY\_FILE, 788  
READ\_BINARY\_FILE\_INTG  
  BINARY\_FILE, 217  
  BINARY\_FILE::READ\_BINARY\_FILE, 788  
READ\_BINARY\_FILE\_INTG1  
  BINARY\_FILE, 217  
  BINARY\_FILE::READ\_BINARY\_FILE, 788  
READ\_BINARY\_FILE\_LINTG  
  BINARY\_FILE, 217  
  BINARY\_FILE::READ\_BINARY\_FILE, 788  
READ\_BINARY\_FILE\_LINTG1  
  BINARY\_FILE, 217  
  BINARY\_FILE::READ\_BINARY\_FILE, 788  
READ\_BINARY\_FILE\_LOGICAL  
  BINARY\_FILE, 218  
  BINARY\_FILE::READ\_BINARY\_FILE, 788  
READ\_BINARY\_FILE\_LOGICAL1  
  BINARY\_FILE, 218  
  BINARY\_FILE::READ\_BINARY\_FILE, 788  
READ\_BINARY\_FILE\_SINTG  
  BINARY\_FILE, 218  
  BINARY\_FILE::READ\_BINARY\_FILE, 789  
READ\_BINARY\_FILE\_SINTG1  
  BINARY\_FILE, 218  
  BINARY\_FILE::READ\_BINARY\_FILE, 789  
READ\_BINARY\_FILE\_SP  
  BINARY\_FILE, 219  
  BINARY\_FILE::READ\_BINARY\_FILE, 789  
READ\_BINARY\_FILE\_SP1  
  BINARY\_FILE, 219  
  BINARY\_FILE::READ\_BINARY\_FILE, 789  
READ\_BINARY\_FILE\_SPC  
  BINARY\_FILE, 219  
  BINARY\_FILE::READ\_BINARY\_FILE, 789  
READ\_BINARY\_FILE\_SPC1  
  BINARY\_FILE, 219  
  BINARY\_FILE::READ\_BINARY\_FILE, 789  
READ\_BINARY\_TAG\_HEADER  
  BINARY\_FILE, 219  
RECEIVE\_BUFFER\_DP  
  TYPES::DISTRIBUTED\_VECTOR\_-  
    TRANSFER\_TYPE, 1083  
RECEIVE\_BUFFER\_INTG  
  TYPES::DISTRIBUTED\_VECTOR\_-  
    TRANSFER\_TYPE, 1083  
RECEIVE\_BUFFER\_L  
  TYPES::DISTRIBUTED\_VECTOR\_-  
    TRANSFER\_TYPE, 1083  
RECEIVE\_BUFFER\_SIZE  
  TYPES::DISTRIBUTED\_VECTOR\_-  
    TRANSFER\_TYPE, 1083  
RECEIVE\_BUFFER\_SP  
  TYPES::DISTRIBUTED\_VECTOR\_-  
    TRANSFER\_TYPE, 1083  
TYPES::DISTRIBUTED\_VECTOR\_-  
  TRANSFER\_TYPE, 1083  
RECEIVE\_TAG\_NUMBER  
  TYPES::DISTRIBUTED\_VECTOR\_-  
    TRANSFER\_TYPE, 1083  
REGION  
  TYPES::DOMAIN\_TYPE, 1120  
  TYPES::EQUATIONS\_SET\_TYPE, 1181  
  TYPES::EQUATIONS\_SETS\_TYPE, 1183  
  TYPES::FIELD\_TYPE, 1227  
  TYPES::FIELD\_VARIABLE\_-  
    COMPONENT\_TYPE, 1231  
  TYPES::FIELD\_VARIABLE\_TYPE, 1235  
  TYPES::FIELDS\_TYPE, 1236  
  TYPES::GENERATED\_MESH\_TYPE, 1240  
  TYPES::MESH\_TYPE, 1270  
  TYPES::MESHS\_TYPE, 1272  
  TYPES::NODES\_TYPE, 1276  
REGION\_COORDINATE\_SYSTEM\_GET  
  REGION\_ROUTINES, 671  
REGION\_COORDINATE\_SYSTEM\_SET\_-  
  NUMBER  
  REGION\_ROUTINES, 672  
  REGION\_ROUTINES::REGION\_-  
    COORDINATE\_SYSTEM\_SET, 1012  
REGION\_COORDINATE\_SYSTEM\_SET\_PTR  
  REGION\_ROUTINES, 672  
  REGION\_ROUTINES::REGION\_-  
    COORDINATE\_SYSTEM\_SET, 1012  
REGION\_CREATE\_FINISH  
  REGION\_ROUTINES, 672  
REGION\_CREATE\_START  
  REGION\_ROUTINES, 672  
REGION\_DESTROY  
  REGION\_ROUTINES, 673  
REGION\_FINISHED  
  TYPES::REGION\_TYPE, 1301  
REGION\_LABEL\_GET  
  REGION\_ROUTINES, 673  
REGION\_LABEL\_SET\_NUMBER  
  REGION\_ROUTINES, 673  
  REGION\_ROUTINES::REGION\_LABEL\_-  
    SET, 1013  
REGION\_LABEL\_SET\_PTR  
  REGION\_ROUTINES, 673  
  REGION\_ROUTINES::REGION\_LABEL\_-  
    SET, 1013  
REGION\_ROUTINES, 671  
  GLOBAL\_REGION, 675  
  REGION\_COORDINATE\_SYSTEM\_GET,  
    671  
REGION\_COORDINATE\_SYSTEM\_SET\_-  
  NUMBER, 672

REGION\_COORDINATE\_SYSTEM\_SET\_-  
     PTR, 672  
 REGION\_CREATE\_FINISH, 672  
 REGION\_CREATE\_START, 672  
 REGION\_DESTROY, 673  
 REGION\_LABEL\_GET, 673  
 REGION\_LABEL\_SET\_NUMBER, 673  
 REGION\_LABEL\_SET\_PTR, 673  
 REGION\_SUB\_REGION\_CREATE\_-  
     FINISH, 674  
 REGION\_SUB\_REGION\_CREATE\_START,  
     674  
 REGION\_USER\_NUMBER\_FIND, 674  
 REGION\_USER\_NUMBER\_FIND\_PTR, 674  
 REGIONS\_FINALISE, 675  
 REGIONS\_INITIALISE, 675  
 REGION\_ROUTINES::REGION\_-  
     COORDINATE\_SYSTEM\_SET, 1012  
 REGION\_COORDINATE\_SYSTEM\_SET\_-  
     NUMBER, 1012  
 REGION\_COORDINATE\_SYSTEM\_SET\_-  
     PTR, 1012  
 REGION\_ROUTINES::REGION\_LABEL\_SET,  
     1013  
     REGION\_LABEL\_SET\_NUMBER, 1013  
     REGION\_LABEL\_SET\_PTR, 1013  
 REGION\_SUB\_REGION\_CREATE\_FINISH  
     REGION\_ROUTINES, 674  
 REGION\_SUB\_REGION\_CREATE\_START  
     REGION\_ROUTINES, 674  
 REGION\_USER\_NUMBER\_FIND  
     REGION\_ROUTINES, 674  
 REGION\_USER\_NUMBER\_FIND\_PTR  
     REGION\_ROUTINES, 674  
 REGIONS\_FINALISE  
     REGION\_ROUTINES, 675  
 REGIONS\_INITIALISE  
     REGION\_ROUTINES, 675  
 REGULAR\_MESH  
     TYPES::GENERATED\_MESH\_TYPE, 1240  
 RELATIVE\_TOLERANCE  
     TYPES::LINEAR\_ITERATIVE\_SOLVER\_-  
         TYPE, 1249  
     TYPES::NONLINEAR\_SOLVER\_TYPE,  
         1281  
 remove\_CH  
     ISO\_VARYING\_STRING, 507  
     ISO\_VARYING\_STRING::remove, 927  
 remove\_VS  
     ISO\_VARYING\_STRING, 507  
     ISO\_VARYING\_STRING::remove, 927  
 repeat\_  
     ISO\_VARYING\_STRING, 507  
     ISO\_VARYING\_STRING::repeat, 928  
 replace\_CH\_CH\_auto  
     ISO\_VARYING\_STRING, 507  
     ISO\_VARYING\_STRING::replace, 929  
 replace\_CH\_CH\_CH\_target  
     ISO\_VARYING\_STRING, 508  
     ISO\_VARYING\_STRING::replace, 929  
 replace\_CH\_CH\_fixed  
     ISO\_VARYING\_STRING, 508  
     ISO\_VARYING\_STRING::replace, 929  
 replace\_CH\_CH\_VS\_target  
     ISO\_VARYING\_STRING, 508  
     ISO\_VARYING\_STRING::replace, 929  
 replace\_CH\_VS\_auto  
     ISO\_VARYING\_STRING, 508  
     ISO\_VARYING\_STRING::replace, 930  
 replace\_CH\_VS\_CH\_target  
     ISO\_VARYING\_STRING, 508  
     ISO\_VARYING\_STRING::replace, 930  
 replace\_CH\_VS\_fixed  
     ISO\_VARYING\_STRING, 508  
     ISO\_VARYING\_STRING::replace, 930  
 replace\_CH\_VS\_VS\_target  
     ISO\_VARYING\_STRING, 508  
     ISO\_VARYING\_STRING::replace, 930  
 replace\_VS\_CH\_auto  
     ISO\_VARYING\_STRING, 509  
     ISO\_VARYING\_STRING::replace, 930  
 replace\_VS\_CH\_CH\_target  
     ISO\_VARYING\_STRING, 509  
     ISO\_VARYING\_STRING::replace, 930  
 replace\_VS\_CH\_fixed  
     ISO\_VARYING\_STRING, 509  
     ISO\_VARYING\_STRING::replace, 930  
 replace\_VS\_CH\_VS\_target  
     ISO\_VARYING\_STRING, 509  
     ISO\_VARYING\_STRING::replace, 931  
 replace\_VS\_VS\_auto  
     ISO\_VARYING\_STRING, 509  
     ISO\_VARYING\_STRING::replace, 931  
 replace\_VS\_VS\_CH\_target  
     ISO\_VARYING\_STRING, 509  
     ISO\_VARYING\_STRING::replace, 931  
 replace\_VS\_VS\_fixed  
     ISO\_VARYING\_STRING, 509  
     ISO\_VARYING\_STRING::replace, 931  
 replace\_VS\_VS\_VS\_target  
     ISO\_VARYING\_STRING, 510  
     ISO\_VARYING\_STRING::replace, 931  
 RESET\_BINARY\_NUMBER\_TAGS  
     BINARY\_FILE, 220  
 RESIDUAL  
     TYPES::EQUATIONS\_MATRICES\_-  
         NONLINEAR\_TYPE, 1153  
     TYPES::SOLVER\_MATRICES\_TYPE, 1323

RESIDUAL\_COEFFICIENT  
  TYPES::EQUATIONS\_MAPPING\_-  
    CREATE\_VALUES\_CACHE\_TYPE,  
      1138  
  TYPES::EQUATIONS\_MAPPING\_-  
    NONLINEAR\_TYPE, 1143  
RESIDUAL\_DOF\_TO\_EQUATIONS\_ROW\_MAP  
  TYPES::EQUATIONS\_MAPPING\_-  
    NONLINEAR\_TYPE, 1143  
RESIDUAL\_VARIABLE  
  TYPES::EQUATIONS\_MAPPING\_-  
    NONLINEAR\_TYPE, 1143  
  TYPES::EQUATIONS\_MAPPING\_TYPE,  
    1150  
RESIDUAL\_VARIABLE\_MAPPING  
  TYPES::EQUATIONS\_MAPPING\_-  
    NONLINEAR\_TYPE, 1143  
  TYPES::EQUATIONS\_MAPPING\_TYPE,  
    1151  
RESIDUAL\_VARIABLE\_TYPE  
  TYPES::EQUATIONS\_MAPPING\_-  
    CREATE\_VALUES\_CACHE\_TYPE,  
      1138  
  TYPES::EQUATIONS\_MAPPING\_-  
    NONLINEAR\_TYPE, 1143  
  TYPES::EQUATIONS\_MAPPING\_TYPE,  
    1151  
  TYPES::SOLUTION\_MAPPING\_CREATE\_-  
    VALUES\_CACHE\_TYPE, 1303  
RHS\_COEFFICIENT  
  TYPES::EQUATIONS\_MAPPING\_-  
    CREATE\_VALUES\_CACHE\_TYPE,  
      1138  
  TYPES::EQUATIONS\_MAPPING\_RHS\_-  
    TYPE, 1146  
RHS\_DOF\_TO\_EQUATIONS\_ROW\_MAP  
  TYPES::EQUATIONS\_MAPPING\_RHS\_-  
    TYPE, 1146  
RHS\_MAPPING  
  TYPES::EQUATIONS\_MAPPING\_TYPE,  
    1151  
RHS\_VARIABLE  
  TYPES::EQUATIONS\_MAPPING\_RHS\_-  
    TYPE, 1146  
RHS\_VARIABLE\_MAPPING  
  TYPES::EQUATIONS\_MAPPING\_RHS\_-  
    TYPE, 1146  
RHS\_VARIABLE\_TYPE  
  TYPES::EQUATIONS\_MAPPING\_-  
    CREATE\_VALUES\_CACHE\_TYPE,  
      1138  
  TYPES::EQUATIONS\_MAPPING\_RHS\_-  
    TYPE, 1146  
TYPES::SOLUTION\_MAPPING\_CREATE\_-  
  VALUES\_CACHE\_TYPE, 1304  
RHS\_VECTOR  
  TYPES::EQUATIONS\_MATRICES\_TYPE,  
    1160  
  TYPES::SOLVER\_MATRICES\_TYPE, 1323  
RIGHT  
  TREES::TREE\_NODE\_TYPE, 1037  
ROOT  
  TREES::TREE\_TYPE, 1038  
ROUTINE\_LIST\_ITEM  
  BASE\_ROUTINES::ROUTINE\_STACK\_-  
    ITEM\_TYPE, 777  
ROUTINE\_STACK  
  BASE\_ROUTINES, 211  
ROW\_DOFS  
  TYPES::ELEMENT\_MATRIX\_TYPE, 1123  
  TYPES::ELEMENT\_VECTOR\_TYPE, 1124  
ROW\_DOFS\_MAPPING  
  TYPES::EQUATIONS\_MAPPING\_TYPE,  
    1151  
  TYPES::SOLUTION\_MAPPING\_TYPE,  
    1307  
ROW\_DOMAIN\_MAPPING  
  TYPES::DISTRIBUTED\_MATRIX\_TYPE,  
    1074  
ROW\_INDICES  
  TYPES::DISTRIBUTED\_MATRIX\_-  
    PETSC\_TYPE, 1072  
  TYPES::MATRIX\_TYPE, 1255  
SAME\_HEADER  
  FIELD\_IO\_ROUTINES::FIELD\_IO\_INFO\_-  
    SET, 847  
SAMEENDIAN  
  binary\_file\_c.c, 1359  
SASUM  
  BLAS::interface, 796  
SAXPY  
  BLAS::interface, 796  
SCALE\_FACTORS  
  TYPES::FIELD\_SCALING\_TYPE, 1222  
SCALING\_INDEX  
  TYPES::FIELD\_VARIABLE\_-  
    COMPONENT\_TYPE, 1231  
SCALING\_TYPE  
  TYPES::FIELD\_SCALINGS\_TYPE, 1224  
SCALINGS  
  TYPES::FIELD\_SCALINGS\_TYPE, 1224  
  TYPES::FIELD\_TYPE, 1228  
scan\_CH\_VS  
  ISO\_VARYING\_STRING, 510  
  ISO\_VARYING\_STRING::scan, 932  
scan\_VS\_CH

ISO\_VARYING\_STRING, 510  
 ISO\_VARYING\_STRING::scan, 932  
 scan\_VS\_VS  
     ISO\_VARYING\_STRING, 510  
     ISO\_VARYING\_STRING::scan, 932  
 SCHEMES  
     TYPES::QUADRATURE\_TYPE, 1298  
 SCOPY  
     BLAS::interface, 796  
 SDOT  
     BLAS::interface, 797  
 SEND\_BUFFER\_DP  
     TYPES::DISTRIBUTED\_VECTOR\_-  
         TRANSFER\_TYPE, 1083  
 SEND\_BUFFER\_INTG  
     TYPES::DISTRIBUTED\_VECTOR\_-  
         TRANSFER\_TYPE, 1084  
 SEND\_BUFFER\_L  
     TYPES::DISTRIBUTED\_VECTOR\_-  
         TRANSFER\_TYPE, 1084  
 SEND\_BUFFER\_SIZE  
     TYPES::DISTRIBUTED\_VECTOR\_-  
         TRANSFER\_TYPE, 1084  
 SEND\_BUFFER\_SP  
     TYPES::DISTRIBUTED\_VECTOR\_-  
         TRANSFER\_TYPE, 1084  
 SEND\_TAG\_NUMBER  
     TYPES::DISTRIBUTED\_VECTOR\_-  
         TRANSFER\_TYPE, 1084  
 SET\_BINARY\_FILE  
     BINARY\_FILE, 220  
 SET\_INDEX  
     TYPES::FIELD\_PARAMETER\_SET\_TYPE,  
         1217  
 SET\_TYPE  
     TYPES::FIELD\_PARAMETER\_SET\_TYPE,  
         1217  
     TYPES::FIELD\_PARAMETER\_SETS\_-  
         TYPE, 1220  
 SHAPE\_SIZE  
     FIELD\_IO\_ROUTINES, 372  
 SHELL\_SORT\_DP  
     SORTING, 724  
     SORTING::SHELL\_SORT, 1016  
 SHELL\_SORT\_INTG  
     SORTING, 724  
     SORTING::SHELL\_SORT, 1016  
 SHELL\_SORT\_SP  
     SORTING, 725  
     SORTING::SHELL\_SORT, 1016  
 SHORT\_INTEGER\_SIZE  
     MACHINE\_CONSTANTS, 542  
 SHORTINTTYPE  
     binary\_file\_c.c, 1359  
 SINGLE\_COMPLEX\_SIZE  
     MACHINE\_CONSTANTS, 542  
 SINGLE\_REAL\_SIZE  
     MACHINE\_CONSTANTS, 542  
 SINTEGER\_SIZE  
     BINARY\_FILE::BINARY\_FILE\_INFO\_-  
         TYPE, 782  
 SINTG  
     KINDS\_IntegerKinds, 94  
 SIZE  
     COMP\_ENVIRONMENT::CACHE\_TYPE,  
         821  
     LISTS::LIST\_TYPE, 958  
     TYPES::MATRIX\_TYPE, 1255  
     TYPES::VECTOR\_TYPE, 1342  
 SKIP\_BINARY\_FILE  
     BINARY\_FILE, 220  
 SKIP\_BINARY\_TAGS  
     BINARY\_FILE, 220  
 SKIP\_CM\_BINARY\_HEADER  
     BINARY\_FILE, 220  
 SNES  
     TYPES::NONLINEAR\_LINESEARCH\_-  
         SOLVER\_TYPE, 1279  
     TYPES::NONLINEAR\_TRUSTREGION\_-  
         SOLVER\_TYPE, 1283  
 SNESCreate  
     CMISS\_PETSC::interface, 805  
 SNESDestroy  
     CMISS\_PETSC::interface, 806  
 SNESGetConvergedReason  
     CMISS\_PETSC::interface, 806  
 SNESGetFunctionNorm  
     CMISS\_PETSC::interface, 806  
 SNESGetIterationNumber  
     CMISS\_PETSC::interface, 806  
 SNESLineSearchSet  
     CMISS\_PETSC::interface, 806  
 SNESLineSearchSetParams  
     CMISS\_PETSC::interface, 806  
 SNESSetFromOptions  
     CMISS\_PETSC::interface, 806  
 SNESSetFunction  
     CMISS\_PETSC::interface, 806  
 SNESSetTolerances  
     CMISS\_PETSC::interface, 806  
 SNESSetTrustRegionTolerance  
     CMISS\_PETSC::interface, 806  
 SNESSetType  
     CMISS\_PETSC::interface, 807  
 SNESSolve  
     CMISS\_PETSC::interface, 807  
 SNRM2  
     BLAS::interface, 797

SOLUTION  
  TYPES::PROBLEM\_LINEAR\_DATA\_TYPE, 1286  
  TYPES::PROBLEM\_NONLINEAR\_DATA\_TYPE, 1287  
  TYPES::PROBLEM\_TIME\_DATA\_TYPE, 1289  
  TYPES::SOLUTION\_MAPPING\_TYPE, 1307  
    TYPES::SOLVER\_TYPE, 1331  
SOLUTION\_FINISHED  
  TYPES::SOLUTION\_TYPE, 1311  
SOLUTION\_MAPPING  
  TYPES::EQUATIONS\_MATRICES\_TYPE, 1160  
  TYPES::EQUATIONS\_SET\_TO\_SOLVER\_MAP\_TYPE, 1178  
  TYPES::SOLUTION\_TYPE, 1311  
  TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SETS\_MAP\_TYPE, 1316  
  TYPES::SOLVER\_MATRICES\_TYPE, 1323  
  TYPES::SOLVER\_TYPE, 1331  
SOLUTION\_MAPPING\_FINISHED  
  TYPES::SOLUTION\_MAPPING\_TYPE, 1307  
SOLUTION\_METHOD  
  TYPES::EQUATIONS\_SET\_TYPE, 1181  
SOLUTION\_NUMBER  
  TYPES::SOLUTION\_TYPE, 1311  
SOLUTION\_TOLERANCE  
  TYPES::NONLINEAR\_SOLVER\_TYPE, 1281  
SOLUTIONS  
  TYPES::PROBLEM\_TYPE, 1291  
SOLVE\_SMALL\_LINEAR\_SYSTEM\_DP  
  MATHS, 551  
  MATHS::SOLVE\_SMALL\_LINEAR\_SYSTEM, 975  
SOLVE\_SMALL\_LINEAR\_SYSTEM\_SP  
  MATHS, 551  
  MATHS::SOLVE\_SMALL\_LINEAR\_SYSTEM, 975  
SOLVE\_TYPE  
  TYPES::SOLVER\_TYPE, 1331  
SOLVER  
  TYPES::EIGENPROBLEM\_SOLVER\_TYPE, 1121  
  TYPES::LINEAR\_SOLVER\_TYPE, 1250  
  TYPES::NONLINEAR\_SOLVER\_TYPE, 1281  
    TYPES::SOLUTION\_TYPE, 1311  
    TYPES::SOLVER\_MATRICES\_TYPE, 1324  
    TYPES::TIME\_INTEGRATION\_SOLVER\_TYPE, 1333  
  SOLVER\_4TH\_RUNGE\_KUTTA\_INTEGRATOR  
    SOLVER\_ROUTINES, 722  
  SOLVER\_ADAMS\_MOULTON\_INTEGRATOR  
    SOLVER\_ROUTINES, 722  
  SOLVER\_CMISS\_LIBRARY  
    SOLVER\_ROUTINES\_SolverLibraries, 132  
  SOLVER\_COL\_TO\_EQUATIONS\_MAPS  
    TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SET\_MAP\_TYPE, 1314  
  SOLVER\_COL\_TO\_EQUATIONS\_SET\_MAPS  
    TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SETS\_MAP\_TYPE, 1316  
  SOLVER\_COL\_TO\_EQUATIONS\_SETS\_MAP  
    TYPES::SOLUTION\_MAPPING\_TYPE, 1307  
  SOLVER\_COLS  
    TYPES::EQUATIONS\_COL\_TO\_SOLVER\_COLS\_MAP\_TYPE, 1125  
    TYPES::JACOBIAN\_COL\_TO\_SOLVER\_COLS\_MAP\_TYPE, 1243  
  SOLVER\_CREATE\_FINISH  
    SOLVER\_ROUTINES, 691  
  SOLVER\_CREATE\_START  
    SOLVER\_ROUTINES, 692  
  SOLVER\_DESTROY  
    SOLVER\_ROUTINES, 692  
  SOLVER\_DIRECT\_CHOLESKY  
    SOLVER\_ROUTINES\_DirectLinearSolverTypes, 136  
  SOLVER\_DIRECT\_LU  
    SOLVER\_ROUTINES\_DirectLinearSolverTypes, 136  
  SOLVER\_DIRECT\_SVD  
    SOLVER\_ROUTINES\_DirectLinearSolverTypes, 136  
  SOLVER\_DOF\_TO\_VARIABLE\_MAPS  
    TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SETS\_MAP\_TYPE, 1316  
  SOLVER\_EIGENPROBLEM\_CREATE\_FINISH  
    SOLVER\_ROUTINES, 693  
  SOLVER\_EIGENPROBLEM\_FINALISE  
    SOLVER\_ROUTINES, 693  
  SOLVER\_EIGENPROBLEM\_INITIALISE  
    SOLVER\_ROUTINES, 694  
  SOLVER\_EIGENPROBLEM\_solve  
    SOLVER\_ROUTINES, 694  
  SOLVER\_EIGENPROBLEM\_TYPE  
    SOLVER\_ROUTINES\_SolverTypes, 130  
  SOLVER\_EULER\_INTEGRATOR  
    SOLVER\_ROUTINES, 722  
  SOLVER\_FINALISE  
    SOLVER\_ROUTINES, 694  
  SOLVER\_FINISHED  
    TYPES::SOLVER\_TYPE, 1332

**SOLVER\_FULL\_MATRICES**  
 SOLVER\_ROUTINES\_SparsityTypes, 152  
**SOLVER\_IMPROVED\_EULER\_INTEGRATOR**  
 SOLVER\_ROUTINES, 722  
**SOLVER\_INITIALISE**  
 SOLVER\_ROUTINES, 695  
**SOLVER\_ITERATIVE\_ADDITIVE\_-**  
 SCHWARZ\_PRECONDITIONER  
 SOLVER\_ROUTINES\_-  
 IterativePreconditionerTypes, 141  
**SOLVER\_ITERATIVE\_BiCGSTAB**  
 SOLVER\_ROUTINES\_-  
 IterativeLinearSolverTypes, 138  
**SOLVER\_ITERATIVE\_BICONJUGATE\_-**  
 GRADIENT  
 SOLVER\_ROUTINES\_-  
 IterativeLinearSolverTypes, 139  
**SOLVER\_ITERATIVE\_BLOCK\_JACOBI\_-**  
 PRECONDITIONER  
 SOLVER\_ROUTINES\_-  
 IterativePreconditionerTypes, 142  
**SOLVER\_ITERATIVE\_CHEBYCHEV**  
 SOLVER\_ROUTINES\_-  
 IterativeLinearSolverTypes, 139  
**SOLVER\_ITERATIVE\_CONJGRAD\_SQUARED**  
 SOLVER\_ROUTINES\_-  
 IterativeLinearSolverTypes, 139  
**SOLVER\_ITERATIVE\_CONJUGATE\_-**  
 GRADIENT  
 SOLVER\_ROUTINES\_-  
 IterativeLinearSolverTypes, 139  
**SOLVER\_ITERATIVE\_GMRES**  
 SOLVER\_ROUTINES\_-  
 IterativeLinearSolverTypes, 140  
**SOLVER\_ITERATIVE\_INCOMPLETE\_-**  
 CHOLESKY\_PRECONDITIONER  
 SOLVER\_ROUTINES\_-  
 IterativePreconditionerTypes, 142  
**SOLVER\_ITERATIVE\_INCOMPLETE\_LU\_-**  
 PRECONDITIONER  
 SOLVER\_ROUTINES\_-  
 IterativePreconditionerTypes, 142  
**SOLVER\_ITERATIVE\_JACOBI\_-**  
 PRECONDITIONER  
 SOLVER\_ROUTINES\_-  
 IterativePreconditionerTypes, 142  
**SOLVER\_ITERATIVE\_NO\_PRECONDITIONER**  
 SOLVER\_ROUTINES\_-  
 IterativePreconditionerTypes, 143  
**SOLVER\_ITERATIVE\_RICHARDSON**  
 SOLVER\_ROUTINES\_-  
 IterativeLinearSolverTypes, 140  
**SOLVER\_ITERATIVE\_SOR\_-**  
 PRECONDITIONER

SOLVER\_ROUTINES\_-  
 IterativePreconditionerTypes, 143  
**SOLVER\_LIBRARY**  
 TYPES::EIGENPROBLEM\_SOLVER\_-  
 TYPE, 1121  
 TYPES::LINEAR\_DIRECT\_SOLVER\_-  
 TYPE, 1246  
 TYPES::LINEAR\_ITERATIVE\_SOLVER\_-  
 TYPE, 1249  
 TYPES::NONLINEAR\_LINESEARCH\_-  
 SOLVER\_TYPE, 1279  
 TYPES::NONLINEAR\_TRUSTREGION\_-  
 SOLVER\_TYPE, 1283  
 TYPES::TIME\_INTEGRATION\_SOLVER\_-  
 TYPE, 1333  
**SOLVER\_LIBRARY\_SET**  
 SOLVER\_ROUTINES, 695  
**SOLVER\_LINEAR\_CREATE\_FINISH**  
 SOLVER\_ROUTINES, 696  
**SOLVER\_LINEAR\_DIRECT\_CREATE\_FINISH**  
 SOLVER\_ROUTINES, 696  
**SOLVER\_LINEAR\_DIRECT\_FINALISE**  
 SOLVER\_ROUTINES, 697  
**SOLVER\_LINEAR\_DIRECT\_INITIALISE**  
 SOLVER\_ROUTINES, 697  
**SOLVER\_LINEAR\_DIRECT\_SOLVE**  
 SOLVER\_ROUTINES, 698  
**SOLVER\_LINEAR\_DIRECT\_SOLVE\_TYPE**  
 SOLVER\_ROUTINES\_LinearSolverTypes,  
 134  
**SOLVER\_LINEAR\_DIRECT\_TYPE\_SET**  
 SOLVER\_ROUTINES, 698  
**SOLVER\_LINEAR\_FINALISE**  
 SOLVER\_ROUTINES, 699  
**SOLVER\_LINEAR\_INITIALISE**  
 SOLVER\_ROUTINES, 699  
**SOLVER\_LINEAR\_ITERATIVE\_ABSOLUTE\_-**  
 TOLERANCE\_SET  
 SOLVER\_ROUTINES, 700  
**SOLVER\_LINEAR\_ITERATIVE\_CREATE\_-**  
 FINISH  
 SOLVER\_ROUTINES, 700  
**SOLVER\_LINEAR\_ITERATIVE\_-**  
 DIVERGENCE\_TOLERANCE\_SET  
 SOLVER\_ROUTINES, 701  
**SOLVER\_LINEAR\_ITERATIVE\_FINALISE**  
 SOLVER\_ROUTINES, 701  
**SOLVER\_LINEAR\_ITERATIVE\_INITIALISE**  
 SOLVER\_ROUTINES, 702  
**SOLVER\_LINEAR\_ITERATIVE\_MAXIMUM\_-**  
 ITERATIONS\_SET  
 SOLVER\_ROUTINES, 702  
**SOLVER\_LINEAR\_ITERATIVE\_-**  
 PRECONDITIONER\_TYPE\_SET

SOLVER\_ROUTINES, 702  
SOLVER\_LINEAR\_ITERATIVE\_RELATIVE\_-  
TOLERANCE\_SET  
SOLVER\_ROUTINES, 703  
SOLVER\_LINEAR\_ITERATIVE\_SOLVE  
SOLVER\_ROUTINES, 703  
SOLVER\_LINEAR\_ITERATIVE\_SOLVE\_TYPE  
SOLVER\_ROUTINES\_LinearSolverTypes,  
134  
SOLVER\_LINEAR\_ITERATIVE\_TYPE\_SET  
SOLVER\_ROUTINES, 704  
SOLVER\_LINEAR\_SOLVE  
SOLVER\_ROUTINES, 704  
SOLVER\_LINEAR\_TYPE  
SOLVER\_ROUTINES\_SolverTypes, 130  
SOLVER\_LINEAR\_TYPE\_SET  
SOLVER\_ROUTINES, 705  
SOLVER\_LSODA\_INTEGRATOR  
SOLVER\_ROUTINES, 722  
SOLVER\_MATRICES  
TYPES::SOLVER\_JACOBIAN\_TYPE, 1320  
TYPES::SOLVER\_MATRIX\_TYPE, 1327  
TYPES::SOLVER\_TYPE, 1332  
SOLVER\_MATRICES\_ASSEMBLE  
SOLVER\_ROUTINES, 705  
SOLVER\_MATRICES\_CREATE\_FINISH  
SOLVER\_MATRICES\_ROUTINES, 677  
SOLVER\_MATRICES\_CREATE\_START  
SOLVER\_MATRICES\_ROUTINES, 677  
SOLVER\_MATRICES\_DESTROY  
SOLVER\_MATRICES\_ROUTINES, 678  
SOLVER\_MATRICES\_FINALISE  
SOLVER\_MATRICES\_ROUTINES, 678  
SOLVER\_MATRICES\_FINISHED  
TYPES::SOLVER\_MATRICES\_TYPE, 1324  
SOLVER\_MATRICES\_INITIALISE  
SOLVER\_MATRICES\_ROUTINES, 679  
SOLVER\_MATRICES\_LIBRARY\_TYPE\_GET  
SOLVER\_MATRICES\_ROUTINES, 679  
SOLVER\_MATRICES\_LIBRARY\_TYPE\_SET  
SOLVER\_MATRICES\_ROUTINES, 679  
SOLVER\_MATRICES\_OUTPUT  
SOLVER\_MATRICES\_ROUTINES, 680  
SOLVER\_MATRICES\_ROUTINES, 676  
SOLVER\_MATRICES\_CREATE\_FINISH,  
677  
SOLVER\_MATRICES\_CREATE\_START,  
677  
SOLVER\_MATRICES\_DESTROY, 678  
SOLVER\_MATRICES\_FINALISE, 678  
SOLVER\_MATRICES\_INITIALISE, 679  
SOLVER\_MATRICES\_LIBRARY\_TYPE\_-  
GET, 679  
SOLVER\_MATRICES\_LIBRARY\_TYPE\_-  
SET, 679  
SOLVER\_MATRICES\_OUTPUT, 680  
SOLVER\_MATRICES\_STORAGE\_TYPE\_-  
GET, 680  
SOLVER\_MATRICES\_STORAGE\_TYPE\_-  
SET, 681  
SOLVER\_MATRIX\_FINALISE, 681  
SOLVER\_MATRIX\_FORM, 682  
SOLVER\_MATRIX\_INITIALISE, 682  
SOLVER\_MATRIX\_STRUCTURE\_-  
CALCULATE, 683  
SOLVER\_MATRICES\_STORAGE\_TYPE\_GET  
SOLVER\_MATRICES\_ROUTINES, 680  
SOLVER\_MATRICES\_STORAGE\_TYPE\_SET  
SOLVER\_MATRICES\_ROUTINES, 681  
SOLVER\_MATRIX  
TYPES::EQUATIONS\_TO\_SOLVER\_-  
MAPS\_TYPE, 1187  
TYPES::JACOBIAN\_TO\_SOLVER\_MAP\_-  
TYPE, 1244  
TYPES::SOLVER\_COL\_TO\_EQUATIONS\_-  
SETS\_MAP\_TYPE, 1316  
SOLVER\_MATRIX\_FINALISE  
SOLVER\_MATRICES\_ROUTINES, 681  
SOLVER\_MATRIX\_FORM  
SOLVER\_MATRICES\_ROUTINES, 682  
SOLVER\_MATRIX\_INITIALISE  
SOLVER\_MATRICES\_ROUTINES, 682  
SOLVER\_MATRIX\_NUMBER  
TYPES::EQUATIONS\_TO\_SOLVER\_-  
MAPS\_TYPE, 1187  
TYPES::EQUATIONS\_TO\_SOLVER\_-  
MATRIX\_MAPS\_SM\_TYPE, 1192  
TYPES::JACOBIAN\_TO\_SOLVER\_MAP\_-  
TYPE, 1244  
TYPES::SOLVER\_COL\_TO\_EQUATIONS\_-  
SETS\_MAP\_TYPE, 1317  
SOLVER\_MATRIX\_OUTPUT  
SOLVER\_ROUTINES\_OutputTypes, 150  
SOLVER\_MATRIX\_STRUCTURE\_-  
CALCULATE  
SOLVER\_MATRICES\_ROUTINES, 683  
SOLVER\_NO\_OUTPUT  
SOLVER\_ROUTINES\_OutputTypes, 150  
SOLVER\_NONLINEAR\_ABSOLUTE\_-  
TOLERANCE\_SET  
SOLVER\_ROUTINES, 706  
SOLVER\_NONLINEAR\_CREATE\_FINISH  
SOLVER\_ROUTINES, 706  
SOLVER\_NONLINEAR\_CUBIC\_LINESEARCH  
SOLVER\_ROUTINES\_-  
NonlinearLineSearchTypes, 146  
SOLVER\_NONLINEAR\_FINALISE

**SOLVER\_ROUTINES**, 707  
**SOLVER\_NONLINEAR\_INITIALISE**  
  **SOLVER\_ROUTINES**, 707  
**SOLVER\_NONLINEAR\_JACOBIAN\_-**  
  **ANALYTIC\_CALCULATED**  
  **SOLVER\_ROUTINES\_-**  
    **JacobianCalculationTypes**, 148  
**SOLVER\_NONLINEAR\_JACOBIAN\_-**  
  **EVALUATE**  
  **SOLVER\_ROUTINES**, 708  
**SOLVER\_NONLINEAR\_JACOBIAN\_-**  
  **EVALUATE\_PETSC**  
  **SOLVER\_ROUTINES**, 708  
**SOLVER\_NONLINEAR\_JACOBIAN\_FD\_-**  
  **CALCULATED**  
  **SOLVER\_ROUTINES\_-**  
    **JacobianCalculationTypes**, 148  
**SOLVER\_NONLINEAR\_JACOBIAN\_NOT\_-**  
  **CALCULATED**  
  **SOLVER\_ROUTINES\_-**  
    **JacobianCalculationTypes**, 149  
**SOLVER\_NONLINEAR\_LINESEARCH**  
  **SOLVER\_ROUTINES\_-**  
    **NonlinearSolverTypes**, 144  
**SOLVER\_NONLINEAR\_LINESEARCH\_-**  
  **ALPHA\_SET**  
  **SOLVER\_ROUTINES**, 708  
**SOLVER\_NONLINEAR\_LINESEARCH\_-**  
  **CREATE\_FINISH**  
  **SOLVER\_ROUTINES**, 709  
**SOLVER\_NONLINEAR\_LINESEARCH\_-**  
  **FINALISE**  
  **SOLVER\_ROUTINES**, 710  
**SOLVER\_NONLINEAR\_LINESEARCH\_-**  
  **INITIALISE**  
  **SOLVER\_ROUTINES**, 710  
**SOLVER\_NONLINEAR\_LINESEARCH\_-**  
  **MAXSTEP\_SET**  
  **SOLVER\_ROUTINES**, 711  
**SOLVER\_NONLINEAR\_LINESEARCH\_SOLVE**  
  **SOLVER\_ROUTINES**, 711  
**SOLVER\_NONLINEAR\_LINESEARCH\_-**  
  **STEPTOL\_SET**  
  **SOLVER\_ROUTINES**, 711  
**SOLVER\_NONLINEAR\_LINESEARCH\_TYPE\_-**  
  **SET**  
  **SOLVER\_ROUTINES**, 712  
**SOLVER\_NONLINEAR\_MAXIMUM\_-**  
  **FUNCTION\_EVALUATIONS\_SET**  
  **SOLVER\_ROUTINES**, 712  
**SOLVER\_NONLINEAR\_MAXIMUM\_-**  
  **ITERATIONS\_SET**  
  **SOLVER\_ROUTINES**, 713  
**SOLVER\_NONLINEAR\_NO\_LINESEARCH**  
  
**SOLVER\_ROUTINES\_-**  
  **NonlinearLineSearchTypes**, 146  
**SOLVER\_NONLINEAR\_NONORMS\_-**  
  **LINESEARCH**  
**SOLVER\_ROUTINES\_-**  
  **NonlinearLineSearchTypes**, 147  
**SOLVER\_NONLINEAR\_QUADRATIC\_-**  
  **LINESEARCH**  
**SOLVER\_ROUTINES\_-**  
  **NonlinearLineSearchTypes**, 147  
**SOLVER\_NONLINEAR\_RELATIVE\_-**  
  **TOLERANCE\_SET**  
  **SOLVER\_ROUTINES**, 713  
**SOLVER\_NONLINEAR\_RESIDUAL\_-**  
  **EVALUATE**  
  **SOLVER\_ROUTINES**, 713  
**SOLVER\_NONLINEAR\_RESIDUAL\_-**  
  **EVALUATE\_PETSC**  
  **SOLVER\_ROUTINES**, 714  
**SOLVER\_NONLINEAR SOLUTION\_-**  
  **TOLERANCE\_SET**  
  **SOLVER\_ROUTINES**, 714  
**SOLVER\_NONLINEAR\_SOLVE**  
  **SOLVER\_ROUTINES**, 715  
**SOLVER\_NONLINEAR\_TRUSTREGION**  
  **SOLVER\_ROUTINES\_-**  
    **NonlinearSolverTypes**, 144  
**SOLVER\_NONLINEAR\_TRUSTREGION\_-**  
  **CREATE\_FINISH**  
  **SOLVER\_ROUTINES**, 715  
**SOLVER\_NONLINEAR\_TRUSTREGION\_-**  
  **DELTA0\_SET**  
  **SOLVER\_ROUTINES**, 716  
**SOLVER\_NONLINEAR\_TRUSTREGION\_-**  
  **FINALISE**  
  **SOLVER\_ROUTINES**, 716  
**SOLVER\_NONLINEAR\_TRUSTREGION\_-**  
  **INITIALISE**  
  **SOLVER\_ROUTINES**, 716  
**SOLVER\_NONLINEAR\_TRUSTREGION\_-**  
  **SOLVE**  
  **SOLVER\_ROUTINES**, 717  
**SOLVER\_NONLINEAR\_TRUSTREGION\_-**  
  **TOLERANCE\_SET**  
  **SOLVER\_ROUTINES**, 717  
**SOLVER\_NONLINEAR\_TYPE**  
  **SOLVER\_ROUTINES\_SolverTypes**, 131  
**SOLVER\_NONLINEAR\_TYPE\_SET**  
  **SOLVER\_ROUTINES**, 718  
**SOLVER\_OUTPUT\_TYPE\_SET**  
  **SOLVER\_ROUTINES**, 718  
**SOLVER\_PETSC\_LIBRARY**  
  **SOLVER\_ROUTINES\_SolverLibraries**, 132  
**SOLVER\_ROUTINES**, 684

SOLVER\_4TH\_RUNGE\_KUTTA\_-  
INTEGRATOR, 722  
SOLVER\_ADAMS\_MOULTON\_-  
INTEGERATOR, 722  
SOLVER\_CREATE\_FINISH, 691  
SOLVER\_CREATE\_START, 692  
SOLVER\_DESTROY, 692  
SOLVER\_EIGENPROBLEM\_CREATE\_-  
FINISH, 693  
SOLVER\_EIGENPROBLEM\_FINALISE,  
693  
SOLVER\_EIGENPROBLEM\_INITIALISE,  
694  
SOLVER\_EIGENPROBLEM\_SOLVE, 694  
SOLVER\_EULER\_INTEGRATOR, 722  
SOLVER\_FINALISE, 694  
SOLVER\_IMPROVED\_EULER\_-  
INTEGRATOR, 722  
SOLVER\_INITIALISE, 695  
SOLVER\_LIBRARY\_SET, 695  
SOLVER\_LINEAR\_CREATE\_FINISH, 696  
SOLVER\_LINEAR\_DIRECT\_CREATE\_-  
FINISH, 696  
SOLVER\_LINEAR\_DIRECT\_FINALISE,  
697  
SOLVER\_LINEAR\_DIRECT\_INITIALISE,  
697  
SOLVER\_LINEAR\_DIRECT\_SOLVE, 698  
SOLVER\_LINEAR\_DIRECT\_TYPE\_SET,  
698  
SOLVER\_LINEAR\_FINALISE, 699  
SOLVER\_LINEAR\_INITIALISE, 699  
SOLVER\_LINEAR\_ITERATIVE\_-  
ABSOLUTE\_TOLERANCE\_SET,  
700  
SOLVER\_LINEAR\_ITERATIVE\_CREATE\_-  
FINISH, 700  
SOLVER\_LINEAR\_ITERATIVE\_-  
DIVERGENCE\_TOLERANCE\_SET,  
701  
SOLVER\_LINEAR\_ITERATIVE\_FINALISE,  
701  
SOLVER\_LINEAR\_ITERATIVE\_-  
INITIALISE, 702  
SOLVER\_LINEAR\_ITERATIVE\_-  
MAXIMUM\_ITERATIONS\_SET,  
702  
SOLVER\_LINEAR\_ITERATIVE\_-  
PRECONDITIONER\_TYPE\_SET,  
702  
SOLVER\_LINEAR\_ITERATIVE\_-  
RELATIVE\_TOLERANCE\_SET,  
703

SOLVER\_LINEAR\_ITERATIVE\_SOLVE,  
703  
SOLVER\_LINEAR\_ITERATIVE\_TYPE\_-  
SET, 704  
SOLVER\_LINEAR\_SOLVE, 704  
SOLVER\_LINEAR\_TYPE\_SET, 705  
SOLVER\_LSODA\_INTEGRATOR, 722  
SOLVER\_MATRICES\_ASSEMBLE, 705  
SOLVER\_NONLINEAR\_ABSOLUTE\_-  
TOLERANCE\_SET, 706  
SOLVER\_NONLINEAR\_CREATE\_FINISH,  
706  
SOLVER\_NONLINEAR\_FINALISE, 707  
SOLVER\_NONLINEAR\_INITIALISE, 707  
SOLVER\_NONLINEAR\_JACOBIAN\_-  
EVALUATE, 708  
SOLVER\_NONLINEAR\_JACOBIAN\_-  
EVALUATE\_PETSC, 708  
SOLVER\_NONLINEAR\_LINESEARCH\_-  
ALPHA\_SET, 708  
SOLVER\_NONLINEAR\_LINESEARCH\_-  
CREATE\_FINISH, 709  
SOLVER\_NONLINEAR\_LINESEARCH\_-  
FINALISE, 710  
SOLVER\_NONLINEAR\_LINESEARCH\_-  
INITIALISE, 710  
SOLVER\_NONLINEAR\_LINESEARCH\_-  
MAXSTEP\_SET, 711  
SOLVER\_NONLINEAR\_LINESEARCH\_-  
SOLVE, 711  
SOLVER\_NONLINEAR\_LINESEARCH\_-  
STEPTOL\_SET, 711  
SOLVER\_NONLINEAR\_LINESEARCH\_-  
TYPE\_SET, 712  
SOLVER\_NONLINEAR\_MAXIMUM\_-  
FUNCTION\_EVALUATIONS\_SET,  
712  
SOLVER\_NONLINEAR\_MAXIMUM\_-  
ITERATIONS\_SET, 713  
SOLVER\_NONLINEAR\_RELATIVE\_-  
TOLERANCE\_SET, 713  
SOLVER\_NONLINEAR\_RESIDUAL\_-  
EVALUATE, 713  
SOLVER\_NONLINEAR\_RESIDUAL\_-  
EVALUATE\_PETSC, 714  
SOLVER\_NONLINEAR SOLUTION\_-  
TOLERANCE\_SET, 714  
SOLVER\_NONLINEAR\_SOLVE, 715  
SOLVER\_NONLINEAR\_TRUSTREGION\_-  
CREATE\_FINISH, 715  
SOLVER\_NONLINEAR\_TRUSTREGION\_-  
DELTAO\_SET, 716  
SOLVER\_NONLINEAR\_TRUSTREGION\_-  
FINALISE, 716

**SOLVER\_NONLINEAR\_TRUSTREGION\_INITIALISE**, 716  
**SOLVER\_NONLINEAR\_TRUSTREGION\_SOLVE**, 717  
**SOLVER\_NONLINEAR\_TRUSTREGION\_TOLERANCE\_SET**, 717  
**SOLVER\_NONLINEAR\_TYPE\_SET**, 718  
**SOLVER\_OUTPUT\_TYPE\_SET**, 718  
**SOLVER\_SOLVE**, 719  
**SOLVER\_SPARSITY\_TYPE\_SET**, 719  
**SOLVER\_TIME\_INTEGRATION\_CREATE\_FINISH**, 720  
**SOLVER\_TIME\_INTEGRATION\_FINALISE**, 720  
**SOLVER\_TIME\_INTEGRATION\_INITIALISE**, 720  
**SOLVER\_TIME\_INTEGRATION\_SOLVE**, 721  
**SOLVER\_VARIABLES\_UPDATE**, 721  
**SOLVER\_ROUTINES::DirectLinearSolverTypes**, 136  
**SOLVER\_ROUTINES::IterativeLinearSolverTypes**, 138  
**SOLVER\_ROUTINES::IterativePreconditionerTypes**, 141  
**SOLVER\_ROUTINES::JacobianCalculationTypes**, 148  
**SOLVER\_ROUTINES::LinearSolverTypes**, 134  
**SOLVER\_ROUTINES::NonlinearLineSearchTypes**, 146  
**SOLVER\_ROUTINES::NonlinearSolverTypes**, 144  
**SOLVER\_ROUTINES::OutputTypes**, 150  
**SOLVER\_ROUTINES::SolverLibraries**, 132  
**SOLVER\_ROUTINES::SolverTypes**, 130  
**SOLVER\_ROUTINES::SparsityTypes**, 152  
**SOLVER\_ROUTINES\_DirectLinearSolverTypes**  
    **SOLVER\_DIRECT\_CHOLESKY**, 136  
    **SOLVER\_DIRECT\_LU**, 136  
    **SOLVER\_DIRECT\_SVD**, 136  
**SOLVER\_ROUTINES\_IterativeLinearSolverTypes**  
    **SOLVER\_ITERATIVE\_BICGSTAB**, 138  
    **SOLVER\_ITERATIVE\_BICONJUGATE\_GRADIENT**, 139  
    **SOLVER\_ITERATIVE\_CHEBYCHEV**, 139  
    **SOLVER\_ITERATIVE\_CONJGRAD\_SQUARED**, 139  
    **SOLVER\_ITERATIVE\_CONJUGATE\_GRADIENT**, 139  
    **SOLVER\_ITERATIVE\_GMRES**, 140  
    **SOLVER\_ITERATIVE\_RICHARDSON**, 140  
**SOLVER\_ROUTINES\_**  
    **IterativePreconditionerTypes**  
    **SOLVER\_ITERATIVE\_ADDITIVE\_SCHWARZ\_PRECONDITIONER**,  
    **SOLVER\_ITERATIVE\_BLOCK\_JACOBI\_PRECONDITIONER**, 142  
    **SOLVER\_ITERATIVE\_INCOMPLETE\_CHOLESKY\_PRECONDITIONER**, 142  
    **SOLVER\_ITERATIVE\_INCOMPLETE\_LU\_PRECONDITIONER**, 142  
    **SOLVER\_ITERATIVE\_JACOBI\_PRECONDITIONER**, 142  
    **SOLVER\_ITERATIVE\_NO\_PRECONDITIONER**, 143  
    **SOLVER\_ITERATIVE\_SOR\_PRECONDITIONER**, 143  
**SOLVER\_ROUTINES\_JacobianCalculationTypes**  
    **SOLVER\_NONLINEAR\_JACOBIAN\_ANALYTIC\_CALCULATED**, 148  
    **SOLVER\_NONLINEAR\_JACOBIAN\_FD\_CALCULATED**, 148  
    **SOLVER\_NONLINEAR\_JACOBIAN\_NOT\_CALCULATED**, 149  
**SOLVER\_ROUTINES\_LinearSolverTypes**  
    **SOLVER\_LINEAR\_DIRECT\_SOLVE\_TYPE**, 134  
    **SOLVER\_LINEAR\_ITERATIVE\_SOLVE\_TYPE**, 134  
**SOLVER\_ROUTINES\_NonlinearLineSearchTypes**  
    **SOLVER\_NONLINEAR\_CUBIC\_LINESEARCH**, 146  
    **SOLVER\_NONLINEAR\_NO\_LINESEARCH**, 146  
    **SOLVER\_NONLINEAR\_NONORMS\_LINESEARCH**, 147  
    **SOLVER\_NONLINEAR\_QUADRATIC\_LINESEARCH**, 147  
**SOLVER\_ROUTINES\_NonlinearSolverTypes**  
    **SOLVER\_NONLINEAR\_LINESearch**, 144  
    **SOLVER\_NONLINEAR\_TRUSTREGION**, 144  
**SOLVER\_ROUTINES\_OutputTypes**  
    **SOLVER\_MATRIX\_OUTPUT**, 150  
    **SOLVER\_NO\_OUTPUT**, 150  
    **SOLVER\_SOLVER\_OUTPUT**, 151  
    **SOLVER\_TIMING\_OUTPUT**, 151  
**SOLVER\_ROUTINES\_SolverLibraries**  
    **SOLVER\_CMISS\_LIBRARY**, 132  
    **SOLVER\_PETSC\_LIBRARY**, 132  
**SOLVER\_ROUTINES\_SolverTypes**  
    **SOLVER\_EIGENPROBLEM\_TYPE**, 130  
    **SOLVER\_LINEAR\_TYPE**, 130  
    **SOLVER\_NONLINEAR\_TYPE**, 131  
    **SOLVER\_TIME\_INTEGRATION\_TYPE**, 131

SOLVER\_ROUTINES\_SparsityTypes  
  SOLVER\_FULL\_MATRICES, 152  
  SOLVER\_SPARSE\_MATRICES, 152

SOLVER\_ROW\_TO\_EQUATIONS\_SET\_MAPS  
  TYPES::SOLUTION\_MAPPING\_TYPE,  
    1307

SOLVER\_ROWS  
  TYPES::EQUATIONS\_ROW\_TO\_-  
    SOLVER\_ROWS\_MAP\_TYPE, 1168

SOLVER\_SOLVE  
  SOLVER\_ROUTINES, 719

SOLVER\_SOLVER\_OUTPUT  
  SOLVER\_ROUTINES\_OutputTypes, 151

SOLVER\_SPARSE\_MATRICES  
  SOLVER\_ROUTINES\_SparsityTypes, 152

SOLVER\_SPARSITY\_TYPE\_SET  
  SOLVER\_ROUTINES, 719

SOLVER\_TIME\_INTEGRATION\_CREATE\_-  
  FINISH  
    SOLVER\_ROUTINES, 720

SOLVER\_TIME\_INTEGRATION\_FINALISE  
  SOLVER\_ROUTINES, 720

SOLVER\_TIME\_INTEGRATION\_INITIALISE  
  SOLVER\_ROUTINES, 720

SOLVER\_TIME\_INTEGRATION\_SOLVE  
  SOLVER\_ROUTINES, 721

SOLVER\_TIME\_INTEGRATION\_TYPE  
  SOLVER\_ROUTINES\_SolverTypes, 131

SOLVER\_TIMING\_OUTPUT  
  SOLVER\_ROUTINES\_OutputTypes, 151

SOLVER\_VARIABLES\_UPDATE  
  SOLVER\_ROUTINES, 721

SOLVER\_VECTOR  
  TYPES::SOLVER\_MATRIX\_TYPE, 1327

SORT\_METHOD  
  LISTS::LIST\_TYPE, 958

SORT\_ORDER  
  LISTS::LIST\_TYPE, 959

SORTING, 723  
  BUBBLE\_SORT\_DP, 723  
  BUBBLE\_SORT\_INTG, 723  
  BUBBLE\_SORT\_SP, 723  
  HEAP\_SORT\_DP, 724  
  HEAP\_SORT\_INTG, 724  
  HEAP\_SORT\_SP, 724  
  SHELL\_SORT\_DP, 724  
  SHELL\_SORT\_INTG, 724  
  SHELL\_SORT\_SP, 725

SORTING::BUBBLE\_SORT, 1014  
  BUBBLE\_SORT\_DP, 1014  
  BUBBLE\_SORT\_INTG, 1014  
  BUBBLE\_SORT\_SP, 1014

SORTING::HEAP\_SORT, 1015  
  HEAP\_SORT\_DP, 1015

HEAP\_SORT\_INTG, 1015  
HEAP\_SORT\_SP, 1015

SORTING::SHELL\_SORT, 1016  
  SHELL\_SORT\_DP, 1016  
  SHELL\_SORT\_INTG, 1016  
  SHELL\_SORT\_SP, 1016

SOURCE  
  TYPES::EQUATIONS\_SET\_TYPE, 1182

SOURCE\_COEFFICIENT  
  TYPES::EQUATIONS\_MAPPING\_-  
    CREATE\_VALUES\_CACHE\_TYPE,  
      1138

TYPES::EQUATIONS\_MAPPING\_-  
  SOURCE\_TYPE, 1148

SOURCE\_DOF\_TO\_EQUATIONS\_ROW\_MAP  
  TYPES::EQUATIONS\_MAPPING\_-  
    SOURCE\_TYPE, 1148

SOURCE\_FIELD  
  TYPES::EQUATIONS\_INTERPOLATION\_-  
    TYPE, 1129

TYPES::EQUATIONS\_SET\_SOURCE\_-  
  TYPE, 1176

SOURCE\_FINISHED  
  TYPES::EQUATIONS\_SET\_SOURCE\_-  
    TYPE, 1176

SOURCE\_INTERP\_PARAMETERS  
  TYPES::EQUATIONS\_INTERPOLATION\_-  
    TYPE, 1129

SOURCE\_INTERP\_POINT  
  TYPES::EQUATIONS\_INTERPOLATION\_-  
    TYPE, 1129

SOURCE\_MAPPING  
  TYPES::EQUATIONS\_MAPPING\_TYPE,  
    1151

SOURCE\_VARIABLE  
  TYPES::EQUATIONS\_MAPPING\_-  
    SOURCE\_TYPE, 1148

SOURCE\_VARIABLE\_MAPPING  
  TYPES::EQUATIONS\_MAPPING\_-  
    SOURCE\_TYPE, 1148

SOURCE\_VARIABLE\_TYPE  
  TYPES::EQUATIONS\_MAPPING\_-  
    CREATE\_VALUES\_CACHE\_TYPE,  
      1138

TYPES::EQUATIONS\_MAPPING\_-  
  SOURCE\_TYPE, 1148

TYPES::SOLUTION\_MAPPING\_CREATE\_-  
  VALUES\_CACHE\_TYPE, 1304

SOURCE\_VECTOR  
  TYPES::EQUATIONS\_MATRICES\_TYPE,  
    1160

SP  
  KINDS\_RealKinds, 95

SP\_FORMAT

BINARY\_FILE::BINARY\_FILE\_INFO\_-  
     TYPE, 782  
 SP\_REAL\_SIZE  
     BINARY\_FILE::BINARY\_FILE\_INFO\_-  
     TYPE, 782  
 SPARSITY\_TYPE  
     TYPES::EQUATIONS\_TYPE, 1196  
     TYPES::SOLVER\_TYPE, 1332  
 SPC  
     KINDS\_ComplexKinds, 99  
 SPC\_REAL\_SIZE  
     BINARY\_FILE::BINARY\_FILE\_INFO\_-  
     TYPE, 782  
 split\_CH  
     ISO\_VARYING\_STRING, 510  
     ISO\_VARYING\_STRING::split, 933  
 split\_VS  
     ISO\_VARYING\_STRING, 510  
     ISO\_VARYING\_STRING::split, 933  
 SROT  
     BLAS::interface, 797  
 SROTG  
     BLAS::interface, 797  
 SSCAL  
     BLAS::interface, 797  
 STACK\_POINTER  
     BASE\_ROUTINES::ROUTINE\_STACK\_-  
     TYPE, 779  
 START\_READ\_COMFILE\_UNIT  
     BASE\_ROUTINES\_FileUnits, 24  
 STOP\_READ\_COMFILE\_UNIT  
     BASE\_ROUTINES\_FileUnits, 25  
 STORAGE\_TYPE  
     TYPES::DISTRIBUTED\_MATRIX\_-  
         PETSC\_TYPE, 1072  
     TYPES::EQUATIONS\_JACOBIAN\_TYPE,  
         1134  
     TYPES::EQUATIONS\_MATRIX\_TYPE,  
         1166  
     TYPES::MATRIX\_TYPE, 1255  
     TYPES::SOLVER\_JACOBIAN\_TYPE, 1320  
     TYPES::SOLVER\_MATRIX\_TYPE, 1327  
 STRING\_TO\_DOUBLE\_C  
     STRINGS, 741  
     STRINGS::STRING\_TO\_DOUBLE, 1028  
 STRING\_TO\_DOUBLE\_VS  
     STRINGS, 741  
     STRINGS::STRING\_TO\_DOUBLE, 1028  
 STRING\_TO\_INTEGER\_C  
     STRINGS, 741  
     STRINGS::STRING\_TO\_INTEGER, 1029  
 STRING\_TO\_INTEGER\_VS  
     STRINGS, 742  
     STRINGS::STRING\_TO\_INTEGER, 1029  
 STRING\_TO\_LOGICAL\_C  
     STRINGS, 742  
     STRINGS::STRING\_TO\_LOGICAL, 1030  
 STRING\_TO\_LOGICAL\_VS  
     STRINGS, 742  
     STRINGS::STRING\_TO\_LOGICAL, 1030  
 STRING\_TO\_LONG\_INTEGER\_C  
     STRINGS, 743  
     STRINGS::STRING\_TO\_LONG\_INTEGER,  
         1031  
 STRING\_TO\_LONG\_INTEGER\_VS  
     STRINGS, 743  
     STRINGS::STRING\_TO\_LONG\_INTEGER,  
         1031  
 STRING\_TO\_MULTI\_INTEGERS\_VS  
     FIELD\_IO\_ROUTINES, 371  
 STRING\_TO\_MULTI\_REALS\_VS  
     FIELD\_IO\_ROUTINES, 371  
 STRING\_TO\_SINGLE\_C  
     STRINGS, 743  
     STRINGS::STRING\_TO\_SINGLE, 1032  
 STRING\_TO\_SINGLE\_VS  
     STRINGS, 744  
     STRINGS::STRING\_TO\_SINGLE, 1032  
 STRINGS, 726  
     CHARACTER\_TO\_LOWERCASE\_C, 730  
     CHARACTER\_TO\_LOWERCASE\_VS, 730  
     CHARACTER\_TO\_UPPERCASE\_C, 730  
     CHARACTER\_TO\_UPPERCASE\_VS, 731  
     IS\_ABBREVIATION\_C\_C, 731  
     IS\_ABBREVIATION\_C\_VS, 731  
     IS\_ABBREVIATION\_VS\_C, 732  
     IS\_ABBREVIATION\_VS\_VS, 732  
     IS\_DIGIT, 732  
     IS\_LETTER, 732  
     IS\_LOWERCASE, 732  
     IS\_UPPERCASE, 733  
     IS\_WHITESPACE, 733  
     LIST\_TO\_CHARACTER\_C, 733  
     LIST\_TO\_CHARACTER\_DP, 734  
     LIST\_TO\_CHARACTER\_INTG, 734  
     LIST\_TO\_CHARACTER\_L, 735  
     LIST\_TO\_CHARACTER\_LINTG, 735  
     LIST\_TO\_CHARACTER\_SP, 736  
     LOGICAL\_TO\_CHARACTER, 736  
     LOGICAL\_TO\_VSTRING, 736  
     NUMBER\_TO\_CHARACTER\_DP, 737  
     NUMBER\_TO\_CHARACTER\_INTG, 737  
     NUMBER\_TO\_CHARACTER\_LINTG, 738  
     NUMBER\_TO\_CHARACTER\_SP, 738  
     NUMBER\_TO\_VSTRING\_DP, 739  
     NUMBER\_TO\_VSTRING\_INTG, 739  
     NUMBER\_TO\_VSTRING\_LINTG, 740  
     NUMBER\_TO\_VSTRING\_SP, 740

STRING\_TO\_DOUBLE\_C, 741  
STRING\_TO\_DOUBLE\_VS, 741  
STRING\_TO\_INTEGER\_C, 741  
STRING\_TO\_INTEGER\_VS, 742  
STRING\_TO\_LOGICAL\_C, 742  
STRING\_TO\_LOGICAL\_VS, 742  
STRING\_TO\_LONG\_INTEGER\_C, 743  
STRING\_TO\_LONG\_INTEGER\_VS, 743  
STRING\_TO\_SINGLE\_C, 743  
STRING\_TO\_SINGLE\_VS, 744  
VSTRING\_TO\_LOWERCASE\_C, 744  
VSTRING\_TO\_LOWERCASE\_VS, 744  
VSTRING\_TO\_UPPERCASE\_C, 745  
VSTRING\_TO\_UPPERCASE\_VS, 745  
STRINGS::CHARACTER\_TO\_LOWERCASE,  
    1017  
        CHARACTER\_TO\_LOWERCASE\_C, 1017  
        CHARACTER\_TO\_LOWERCASE\_VS, 1017  
STRINGS::CHARACTER\_TO\_UPPERCASE,  
    1018  
        CHARACTER\_TO\_UPPERCASE\_C, 1018  
        CHARACTER\_TO\_UPPERCASE\_VS, 1018  
STRINGS::IS\_ABBREVIATION, 1019  
    IS\_ABBREVIATION\_C\_C, 1019  
    IS\_ABBREVIATION\_C\_VS, 1019  
    IS\_ABBREVIATION\_VS\_C, 1019  
    IS\_ABBREVIATION\_VS\_VS, 1020  
STRINGS::LIST\_TO\_CHARACTER, 1021  
    LIST\_TO\_CHARACTER\_C, 1021  
    LIST\_TO\_CHARACTER\_DP, 1021  
    LIST\_TO\_CHARACTER\_INTG, 1022  
    LIST\_TO\_CHARACTER\_L, 1022  
    LIST\_TO\_CHARACTER\_LINTG, 1022  
    LIST\_TO\_CHARACTER\_SP, 1023  
STRINGS::NUMBER\_TO\_CHARACTER, 1024  
    NUMBER\_TO\_CHARACTER\_DP, 1024  
    NUMBER\_TO\_CHARACTER\_INTG, 1024  
    NUMBER\_TO\_CHARACTER\_LINTG, 1024  
    NUMBER\_TO\_CHARACTER\_SP, 1025  
STRINGS::NUMBER\_TO\_VSTRING, 1026  
    NUMBER\_TO\_VSTRING\_DP, 1026  
    NUMBER\_TO\_VSTRING\_INTG, 1026  
    NUMBER\_TO\_VSTRING\_LINTG, 1026  
    NUMBER\_TO\_VSTRING\_SP, 1027  
STRINGS::STRING\_TO\_DOUBLE, 1028  
    STRING\_TO\_DOUBLE\_C, 1028  
    STRING\_TO\_DOUBLE\_VS, 1028  
STRINGS::STRING\_TO\_INTEGER, 1029  
    STRING\_TO\_INTEGER\_C, 1029  
    STRING\_TO\_INTEGER\_VS, 1029  
STRINGS::STRING\_TO\_LOGICAL, 1030  
    STRING\_TO\_LOGICAL\_C, 1030  
    STRING\_TO\_LOGICAL\_VS, 1030  
STRINGS::STRING\_TO\_LONG\_INTEGER, 1031  
    STRING\_TO\_LONG\_INTEGER\_C, 1031  
    STRING\_TO\_LONG\_INTEGER\_VS, 1031  
STRINGS::STRING\_TO\_SINGLE, 1032  
    STRING\_TO\_SINGLE\_C, 1032  
    STRING\_TO\_SINGLE\_VS, 1032  
STRINGS::VSTRING\_TO\_LOWERCASE, 1033  
    VSTRING\_TO\_LOWERCASE\_C, 1033  
    VSTRING\_TO\_LOWERCASE\_VS, 1033  
STRINGS::VSTRING\_TO\_UPPERCASE, 1034  
    VSTRING\_TO\_UPPERCASE\_C, 1034  
    VSTRING\_TO\_UPPERCASE\_VS, 1034  
STRSV  
    BLAS::interface, 797  
STRUCTURE\_TYPE  
    TYPES::EQUATIONS\_JACOBIAN\_TYPE,  
        1134  
    TYPES::EQUATIONS\_MATRIX\_TYPE,  
        1166  
SUB\_BASES  
    TYPES::BASIS\_TYPE, 1051  
SUB\_REGIONS  
    TYPES::REGION\_TYPE, 1302  
SUBTYPE  
    TYPES::EQUATIONS\_SET\_TYPE, 1182  
    TYPES::PROBLEM\_TYPE, 1291  
SURROUNDING\_ELEMENTS  
    TYPES::DECOMPOSITION\_LINE\_TYPE,  
        1060  
    TYPES::DOMAIN\_NODE\_TYPE, 1112  
    TYPES::MESH\_NODE\_TYPE, 1262  
SYSTEM\_CPU  
    TIMER, 746  
    timer\_c.c, 1534  
TEMPORARY\_FILE\_UNIT  
    BASE\_ROUTINES\_FileUnits, 25  
TIME\_DATA  
    TYPES::EQUATIONS\_TYPE, 1196  
TIME\_INTEGRATION\_SOLVER  
    TYPES::SOLVER\_TYPE, 1332  
TIME\_TYPE  
    TYPES::EQUATIONS\_SET\_TYPE, 1182  
TIMER, 746  
    CPU\_TIMER, 746  
    SYSTEM\_CPU, 746  
    TOTAL\_CPU, 746  
    USER\_CPU, 746  
    TIMER::interface, 1035  
    CPUTIMER, 1035  
    timer\_c.c  
        CPUTimer, 1534  
        SYSTEM\_CPU, 1534  
        TOTAL\_CPU, 1534  
        USER\_CPU, 1534

**TIMING**  
 BASE\_ROUTINES, 211  
 BASE\_ROUTINES::ROUTINE\_STACK\_-  
     ITEM\_TYPE, 777  
**TIMING\_ALL\_SUBROUTINES**  
 BASE\_ROUTINES, 211  
**TIMING\_FILE\_OPEN**  
 BASE\_ROUTINES, 211  
**TIMING\_FILE\_UNIT**  
 BASE\_ROUTINES\_FileUnits, 25  
**TIMING\_FROM\_SUBROUTINE**  
 BASE\_ROUTINES, 212  
**TIMING\_OUTPUT\_TYPE**  
 BASE\_ROUTINES\_OutputType, 21  
**TIMING\_ROUTINE\_LIST**  
 BASE\_ROUTINES, 212  
**TIMING\_SET\_OFF**  
 BASE\_ROUTINES, 206  
**TIMING\_SET\_ON**  
 BASE\_ROUTINES, 206  
**TIMING\_SUMMARY**  
 BASE\_ROUTINES, 212  
**TIMING\_SUMMARY\_OUTPUT**  
 BASE\_ROUTINES, 206  
**TOPOLOGY**  
 TYPES::DECOMPOSITION\_TYPE, 1066  
 TYPES::DOMAIN\_TYPE, 1120  
 TYPES::MESH\_TYPE, 1271  
**TOTAL\_CPU**  
 TIMER, 746  
 timer\_c.c, 1534  
**TOTAL\_EXCLUSIVE\_CPU\_TIME**  
 BASE\_ROUTINES::ROUTINE\_LIST\_-  
     ITEM\_TYPE, 774  
**TOTAL\_EXCLUSIVE\_SYSTEM\_TIME**  
 BASE\_ROUTINES::ROUTINE\_LIST\_-  
     ITEM\_TYPE, 774  
**TOTAL\_INCLUSIVE\_CPU\_TIME**  
 BASE\_ROUTINES::ROUTINE\_LIST\_-  
     ITEM\_TYPE, 774  
**TOTAL\_INCLUSIVE\_SYSTEM\_TIME**  
 BASE\_ROUTINES::ROUTINE\_LIST\_-  
     ITEM\_TYPE, 774  
**TOTAL\_NUMBER\_OF\_DOFS**  
 TYPES::DOMAIN\_DOFS\_TYPE, 1089  
 TYPES::FIELD\_VARIABLE\_TYPE, 1235  
 TYPES::SOLVER\_COL\_TO\_EQUATIONS\_-  
     SETS\_MAP\_TYPE, 1317  
**TOTAL\_NUMBER\_OF\_ELEMENTS**  
 TYPES::DECOMPOSITION\_ELEMENTS\_-  
     TYPE, 1057  
 TYPES::DOMAIN\_ELEMENTS\_TYPE,  
     1094  
**TOTAL\_NUMBER\_OF\_LOCAL**

TYPES::DOMAIN\_MAPPING\_TYPE, 1108  
**TOTAL\_NUMBER\_OF\_NODES**  
 TYPES::DOMAIN\_NODES\_TYPE, 1115  
**TOTAL\_NUMBER\_OF\_ROWS**  
 TYPES::EQUATIONS\_MAPPING\_TYPE,  
     1151  
 TYPES::EQUATIONS\_MATRICES\_TYPE,  
     1161  
**TRANSFERS**  
 TYPES::DISTRIBUTED\_VECTOR\_-  
     CMISS\_TYPE, 1078  
**TREE\_BLACK\_NODE**  
 TREES\_TreeNodeColourTypes, 154  
**TREE\_CREATE\_FINISH**  
 TREES, 750  
**TREE\_CREATE\_START**  
 TREES, 750  
**TREE\_DESTROY**  
 TREES, 750  
**TREE\_DETACH\_AND\_DESTROY**  
 TREES, 751  
**TREE\_DETACH\_IN\_ORDER**  
 TREES, 751  
**TREE\_DUPLICATES\_ALLOWED\_TYPE**  
 TREES\_TreeInsertTypes, 156  
**TREE\_FINALISE**  
 TREES, 752  
**TREE\_FINISHED**  
 TREES::TREE\_TYPE, 1038  
**TREE\_INITIALISE**  
 TREES, 752  
**TREE\_INSERT\_TYPE\_SET**  
 TREES, 753  
**TREE\_ITEM\_DELETE**  
 TREES, 753  
**TREE\_ITEM\_INSERT**  
 TREES, 754  
**TREE\_NO\_DUPLICATES\_ALLOWED**  
 TREES\_TreeInsertTypes, 156  
**TREE\_NODE\_DUPLICATE\_KEY**  
 TREES\_TreeNodeInsertStatus, 155  
**TREE\_NODE\_FINALISE**  
 TREES, 754  
**TREE\_NODE\_INITIALISE**  
 TREES, 755  
**TREE\_NODE\_INSERT\_SUCESSFUL**  
 TREES\_TreeNodeInsertStatus, 155  
**TREE\_NODE\_KEY\_GET**  
 TREES, 755  
**TREE\_NODE\_VALUE\_GET**  
 TREES, 755  
**TREE\_NODE\_VALUE\_SET**  
 TREES, 756  
**TREE\_OUTPUT**

TREES, 756  
TREE\_OUTPUT\_IN\_ORDER  
    TREES, 757  
TREE\_PREDECESSOR  
    TREES, 757  
TREE\_RED\_NODE  
    TREES\_TreeNodeColourTypes, 154  
TREE\_SEARCH  
    TREES, 758  
TREE\_SUCCESSOR  
    TREES, 758  
TREES, 748  
    TREE\_CREATE\_FINISH, 750  
    TREE\_CREATE\_START, 750  
    TREE\_DESTROY, 750  
    TREE\_DETACH\_AND\_DESTROY, 751  
    TREE\_DETACH\_IN\_ORDER, 751  
    TREE\_FINALISE, 752  
    TREE\_INITIALISE, 752  
    TREE\_INSERT\_TYPE\_SET, 753  
    TREE\_ITEM\_DELETE, 753  
    TREE\_ITEM\_INSERT, 754  
    TREE\_NODE\_FINALISE, 754  
    TREE\_NODE\_INITIALISE, 755  
    TREE\_NODE\_KEY\_GET, 755  
    TREE\_NODE\_VALUE\_GET, 755  
    TREE\_NODE\_VALUE\_SET, 756  
    TREE\_OUTPUT, 756  
    TREE\_OUTPUT\_IN\_ORDER, 757  
    TREE\_PREDECESSOR, 757  
    TREE\_SEARCH, 758  
    TREE\_SUCCESSOR, 758  
TREES::TREE\_NODE\_TYPE, 1036  
    COLOUR, 1036  
    KEY, 1036  
    LEFT, 1036  
    PARENT, 1036  
    RIGHT, 1037  
    VALUE, 1037  
TREES::TREE\_TYPE, 1038  
    INSERT\_TYPE, 1038  
    NIL, 1038  
    NUMBER\_IN\_TREE, 1038  
    ROOT, 1038  
        TREE\_FINISHED, 1038  
TREES::TreeInsertTypes, 156  
TREES::TreeNodeColourTypes, 154  
TREES::TreeNodeInsertStatus, 155  
TREES\_TreeInsertTypes  
    TREE\_DUPLICATES\_ALLOWED\_TYPE,  
        156  
        TREE\_NO\_DUPLICATES\_ALLOWED, 156  
TREES\_TreeNodeColourTypes  
    TREE\_BLACK\_NODE, 154  
    TREE\_RED\_NODE, 154  
TREES\_TreeNodeInsertStatus  
    TREE\_NODE\_DUPLICATE\_KEY, 155  
    TREE\_NODE\_INSERT\_SUCESSFUL, 155  
trim\_  
    ISO\_VARYING\_STRING, 510  
    ISO\_VARYING\_STRING::trim, 934  
TRUSTREGION\_DELTA0  
    TYPES::NONLINEAR\_TRUSTREGION\_-  
        SOLVER\_TYPE, 1284  
TRUSTREGION\_SOLVER  
    TYPES::NONLINEAR\_SOLVER\_TYPE,  
        1281  
TRUSTREGION\_TOLERANCE  
    TYPES::NONLINEAR\_TRUSTREGION\_-  
        SOLVER\_TYPE, 1284  
TYPE  
    TYPES::BASIS\_TYPE, 1051  
    TYPES::COORDINATE\_SYSTEM\_TYPE,  
        1053  
    TYPES::EQUATIONS\_SET\_TYPE, 1182  
    TYPES::FIELD\_TYPE, 1228  
    TYPES::PROBLEM\_TYPE, 1291  
    TYPES::QUADRATURE\_TYPE, 1298  
TYPES, 760  
    COMP\_ENVIRONMENT::MPI\_-  
        COMPUTATIONAL\_NODE\_TYPE,  
            827  
TYPES::BASIS\_FUNCTIONS\_TYPE, 1040  
    BASES, 1040  
        NUMBER\_BASIS\_FUNCTIONS, 1040  
TYPES::BASIS\_PTR\_TYPE, 1041  
    PTR, 1041  
TYPES::BASIS\_TYPE, 1042  
    BASIS\_FINISHED, 1045  
    COLLAPSED\_XI, 1045  
    DEGENERATE, 1045  
    DERIVATIVE\_NUMBERS\_IN\_LOCAL\_-  
        LINE, 1045  
    DERIVATIVE\_ORDER\_INDEX, 1046  
    DERIVATIVE\_ORDER\_INDEX\_INV, 1046  
    ELEMENT\_PARAMETER\_INDEX, 1046  
    FACE\_BASES, 1046  
    FAMILY\_NUMBER, 1046  
    GLOBAL\_NUMBER, 1047  
    HERMITE, 1047  
    INTERPOLATION\_ORDER, 1047  
    INTERPOLATION\_TYPE, 1047  
    INTERPOLATION\_XI, 1047  
    LINE\_BASES, 1047  
    LOCAL\_LINE\_XI\_DIRECTION, 1048  
    MAXIMUM\_NUMBER\_OF\_-  
        DERIVATIVES, 1048  
    NODE\_AT\_COLLAPSE, 1048

NODE\_NUMBERS\_IN\_LOCAL\_LINE, 1048  
 NODE\_POSITION\_INDEX, 1048  
 NODE\_POSITION\_INDEX\_INV, 1048  
 NUMBER\_OF\_COLLAPSED\_XI, 1049  
 NUMBER\_OF\_DERIVATIVES, 1049  
 NUMBER\_OF\_ELEMENT\_PARAMETERS,  
     1049  
 NUMBER\_OF\_LOCAL\_LINES, 1049  
 NUMBER\_OF\_NODES, 1049  
 NUMBER\_OF\_NODES\_IN\_LOCAL\_LINE,  
     1049  
 NUMBER\_OF\_NODES\_XI, 1049  
 NUMBER\_OF\_PARTIAL\_DERIVATIVES,  
     1050  
 NUMBER\_OF\_SUB\_BASES, 1050  
 NUMBER\_OF\_XI, 1050  
 NUMBER\_OF\_XI\_COORDINATES, 1050  
 PARENT\_BASIS, 1050  
 PARTIAL\_DERIVATIVE\_INDEX, 1050  
 QUADRATURE, 1050  
 SUB\_BASES, 1051  
 TYPE, 1051  
 USER\_NUMBER, 1051  
 TYPES::COORDINATE\_SYSTEM\_TYPE, 1052  
     COORDINATE\_SYSTEM\_FINISHED, 1053  
     FOCUS, 1053  
     NUMBER\_OF\_DIMENSIONS, 1053  
     ORIENTATION, 1053  
     ORIGIN, 1053  
     RADIAL\_INTERPOLATION\_TYPE, 1053  
     TYPE, 1053  
     USER\_NUMBER, 1054  
 TYPES::DECOMPOSITION\_ELEMENT\_TYPE,  
     1055  
     ADJACENT\_ELEMENTS, 1055  
     ELEMENT\_LINES, 1055  
     GLOBAL\_NUMBER, 1056  
     LOCAL\_NUMBER, 1056  
     NUMBER\_OF\_ADJACENT\_ELEMENTS,  
         1056  
     USER\_NUMBER, 1056  
 TYPES::DECOMPOSITION\_ELEMENTS\_-  
     TYPE, 1057  
     DECOMPOSITION, 1057  
     ELEMENTS, 1057  
     TOTAL\_NUMBER\_OF\_ELEMENTS, 1057  
 TYPES::DECOMPOSITION\_LINE\_TYPE, 1059  
     ADJACENT\_LINES, 1059  
     ELEMENT\_LINES, 1059  
     NUMBER, 1059  
     NUMBER\_OF\_SURROUNDING\_-  
         ELEMENTS, 1060  
     SURROUNDING\_ELEMENTS, 1060  
     XI\_DIRECTION, 1060  
 TYPES::DECOMPOSITION\_LINES\_TYPE, 1061  
     DECOMPOSITION, 1061  
     LINES, 1061  
     NUMBER\_OF\_LINES, 1061  
 TYPES::DECOMPOSITION\_PTR\_TYPE, 1062  
     PTR, 1062  
 TYPES::DECOMPOSITION\_TOPOLOGY\_-  
     TYPE, 1063  
     DECOMPOSITION, 1063  
     ELEMENTS, 1063  
     LINES, 1063  
 TYPES::DECOMPOSITION\_TYPE, 1064  
     DECOMPOSITION\_FINISHED, 1065  
     DECOMPOSITION\_TYPE, 1065  
     DECOMPOSITIONS, 1065  
     DOMAIN, 1065  
     ELEMENT\_DOMAIN, 1065  
     GLOBAL\_NUMBER, 1065  
     MESH, 1066  
     MESH\_COMPONENT\_NUMBER, 1066  
     NUMBER\_OF\_DOMAINS, 1066  
     NUMBER\_OF\_EDGES\_CUT, 1066  
     TOPOLOGY, 1066  
     USER\_NUMBER, 1066  
 TYPES::DECOMPOSITIONS\_TYPE, 1067  
     DECOMPOSITIONS, 1067  
     MESH, 1067  
     NUMBER\_OF\_DECOMPOSITIONS, 1067  
 TYPES::DISTRIBUTED\_MATRIX\_CMISS\_-  
     TYPE, 1068  
     BASE\_TAG\_NUMBER, 1068  
     DISTRIBUTED\_MATRIX, 1068  
     MATRIX, 1068  
 TYPES::DISTRIBUTED\_MATRIX\_PETSC\_-  
     TYPE, 1069  
     COLUMN\_INDICES, 1070  
     DATA\_DP, 1070  
     DATA\_SIZE, 1070  
     DIAGONAL\_NUMBER\_NON\_ZEROS, 1070  
     DISTRIBUTED\_MATRIX, 1070  
     GLOBAL\_M, 1071  
     GLOBAL\_N, 1071  
     GLOBAL\_ROW\_NUMBERS, 1071  
     ISLTGMAPPING, 1071  
     M, 1071  
     MATRIX, 1071  
     MAXIMUM\_COLUMN\_INDICES\_PER\_-  
         ROW, 1071  
     N, 1071  
     NUMBER\_NON\_ZEROS, 1072  
     OFFDIAGONAL\_NUMBER\_NON\_ZEROS,  
         1072  
     ROW\_INDICES, 1072  
     STORAGE\_TYPE, 1072

TYPES::DISTRIBUTED\_MATRIX\_TYPE, 1073  
CMISS, 1073  
COLUMN\_DOMAIN\_MAPPING, 1073  
DATA\_TYPE, 1074  
GHOSTING\_TYPE, 1074  
LIBRARY\_TYPE, 1074  
MATRIX\_FINISHED, 1074  
PETSC, 1074  
ROW\_DOMAIN\_MAPPING, 1074  
TYPES::DISTRIBUTED\_VECTOR\_CMISS\_-  
TYPE, 1076  
BASE\_TAG\_NUMBER, 1077  
DATA\_DP, 1077  
DATA\_INTG, 1077  
DATA\_L, 1077  
DATA\_SIZE, 1077  
DATA\_SP, 1077  
DISTRIBUTED\_VECTOR, 1077  
N, 1077  
TRANSFERS, 1078  
TYPES::DISTRIBUTED\_VECTOR\_PETSC\_-  
TYPE, 1079  
DATA\_SIZE, 1079  
DISTRIBUTED\_VECTOR, 1079  
GLOBAL\_N, 1079  
GLOBAL\_NUMBERS, 1080  
ISLTGMAPPING, 1080  
N, 1080  
VECTOR, 1080  
TYPES::DISTRIBUTED\_VECTOR\_-  
TRANSFER\_TYPE, 1081  
CMISS\_VECTOR, 1082  
DATA\_TYPE, 1082  
MPI\_RECEIVE\_REQUEST, 1082  
MPI\_SEND\_REQUEST, 1082  
RECEIVE\_BUFFER\_DP, 1083  
RECEIVE\_BUFFER\_INTG, 1083  
RECEIVE\_BUFFER\_L, 1083  
RECEIVE\_BUFFER\_SIZE, 1083  
RECEIVE\_BUFFER\_SP, 1083  
RECEIVE\_TAG\_NUMBER, 1083  
SEND\_BUFFER\_DP, 1083  
SEND\_BUFFER\_INTG, 1084  
SEND\_BUFFER\_L, 1084  
SEND\_BUFFER\_SIZE, 1084  
SEND\_BUFFER\_SP, 1084  
SEND\_TAG\_NUMBER, 1084  
TYPES::DISTRIBUTED\_VECTOR\_TYPE, 1085  
CMISS, 1085  
DATA\_TYPE, 1085  
DOMAIN\_MAPPING, 1086  
GHOSTING\_TYPE, 1086  
LIBRARY\_TYPE, 1086  
PETSC, 1086  
VECTOR\_FINISHED, 1086  
TYPES::DOMAIN\_ADJACENT\_DOMAIN\_-  
TYPE, 1087  
DOMAIN\_NUMBER, 1087  
LOCAL\_GHOST\_RECEIVE\_INDICES,  
1087  
LOCAL\_GHOST\_SEND\_INDICES, 1087  
NUMBER\_OF\_RECEIVE\_GHOSTS, 1088  
NUMBER\_OF\_SEND\_GHOSTS, 1088  
TYPES::DOMAIN\_DOFS\_TYPE, 1089  
DOF\_INDEX, 1089  
DOMAIN, 1089  
NUMBER\_OF\_DOFS, 1089  
TOTAL\_NUMBER\_OF\_DOFS, 1089  
TYPES::DOMAIN\_ELEMENT\_TYPE, 1091  
BASIS, 1091  
ELEMENT\_DERIVATIVES, 1091  
ELEMENT\_NODES, 1091  
NUMBER, 1091  
TYPES::DOMAIN\_ELEMENTS\_TYPE, 1093  
DOMAIN, 1093  
ELEMENTS, 1093  
MAXIMUM\_NUMBER\_OF\_ELEMENT\_-  
PARAMETERS, 1093  
NUMBER\_OF\_ELEMENTS, 1094  
TOTAL\_NUMBER\_OF\_ELEMENTS, 1094  
TYPES::DOMAIN\_FACE\_PTR\_TYPE, 1095  
PTR, 1095  
TYPES::DOMAIN\_FACE\_TYPE, 1096  
BASIS, 1096  
DERIVATIVES\_IN\_FACE, 1096  
NODES\_IN\_FACE, 1096  
NUMBER, 1097  
XI\_DIRECTION1, 1097  
XI\_DIRECTION2, 1097  
TYPES::DOMAIN\_FACES\_TYPE, 1098  
DOMAIN, 1098  
FACES, 1098  
NUMBER\_OF\_FACES, 1098  
TYPES::DOMAIN\_GLOBAL\_MAPPING\_TYPE,  
1099  
DOMAIN\_NUMBER, 1099  
LOCAL\_NUMBER, 1099  
LOCAL\_TYPE, 1099  
NUMBER\_OF\_DOMAINS, 1100  
TYPES::DOMAIN\_LINE\_PTR\_TYPE, 1101  
PTR, 1101  
TYPES::DOMAIN\_LINE\_TYPE, 1102  
BASIS, 1102  
DERIVATIVES\_IN\_LINE, 1102  
NODES\_IN\_LINE, 1102  
NUMBER, 1102  
TYPES::DOMAIN\_LINES\_TYPE, 1104  
DOMAIN, 1104

LINES, 1104  
 NUMBER\_OF\_LINES, 1104

**TYPES::DOMAIN\_MAPPING\_TYPE**, 1105  
 ADJACENT\_DOMAINS, 1106  
 ADJACENT\_DOMAINS\_LIST, 1106  
 ADJACENT\_DOMAINS\_PTR, 1106  
 BOUNDARY\_LIST, 1107  
 GHOST\_LIST, 1107  
 GLOBAL\_TO\_LOCAL\_MAP, 1107  
 INTERNAL\_LIST, 1107  
 LOCAL\_TO\_GLOBAL\_MAP, 1107  
 NUMBER\_OF\_ADJACENT\_DOMAINS,  
   1107  
 NUMBER\_OF\_BOUNDARY, 1107  
 NUMBER\_OF\_DOMAIN\_LOCAL, 1108  
 NUMBER\_OF\_DOMAINS, 1108  
 NUMBER\_OF\_GHOST, 1108  
 NUMBER\_OF\_GLOBAL, 1108  
 NUMBER\_OF\_INTERNAL, 1108  
 NUMBER\_OF\_LOCAL, 1108  
 TOTAL\_NUMBER\_OF\_LOCAL, 1108

**TYPES::DOMAIN\_MAPPINGS\_TYPE**, 1109  
 DOFS, 1109  
 DOMAIN, 1109  
 ELEMENTS, 1109  
 NODES, 1109

**TYPES::DOMAIN\_NODE\_TYPE**, 1111  
 DOF\_INDEX, 1112  
 GLOBAL\_NUMBER, 1112  
 LOCAL\_NUMBER, 1112  
 NODE\_LINES, 1112  
 NUMBER\_OF\_DERIVATIVES, 1112  
 NUMBER\_OF\_NODE\_LINES, 1112  
 NUMBER\_OF\_SURROUNDING\_-  
   ELEMENTS, 1112  
 PARTIAL\_DERIVATIVE\_INDEX, 1112  
 SURROUNDING\_ELEMENTS, 1112  
 USER\_NUMBER, 1113

**TYPES::DOMAIN\_NODES\_TYPE**, 1114  
 DOMAIN, 1114  
 MAXIMUM\_NUMBER\_OF\_-  
   DERIVATIVES, 1114  
 NODES, 1114  
 NUMBER\_OF\_NODES, 1115  
 TOTAL\_NUMBER\_OF\_NODES, 1115

**TYPES::DOMAIN\_PTR\_TYPE**, 1116  
 PTR, 1116

**TYPES::DOMAIN\_TOPOLOGY\_TYPE**, 1117  
 DOFS, 1117  
 DOMAIN, 1117  
 ELEMENTS, 1117  
 FACES, 1118  
 LINES, 1118  
 NODES, 1118

TYPES::DOMAIN\_TYPE, 1119  
 DECOMPOSITION, 1119  
 MAPPINGS, 1119  
 MESH, 1120  
 MESH\_COMPONENT\_NUMBER, 1120  
 NODE\_DOMAIN, 1120  
 NUMBER\_OF\_DIMENSIONS, 1120  
 REGION, 1120  
 TOPOLOGY, 1120

**TYPES::EIGENPROBLEM\_SOLVER\_TYPE**,  
   1121  
 SOLVER, 1121  
 SOLVER\_LIBRARY, 1121

**TYPES::ELEMENT\_MATRIX\_TYPE**, 1122  
 COLUMN\_DOFS, 1122  
 EQUATIONS\_MATRIX\_NUMBER, 1122  
 MATRIX, 1122  
 MAX\_NUMBER\_OF\_COLUMNS, 1122  
 MAX\_NUMBER\_OF\_ROWS, 1122  
 NUMBER\_OF\_COLUMNS, 1122  
 NUMBER\_OF\_ROWS, 1123  
 ROW\_DOFS, 1123

**TYPES::ELEMENT\_VECTOR\_TYPE**, 1124  
 MAX\_NUMBER\_OF\_ROWS, 1124  
 NUMBER\_OF\_ROWS, 1124  
 ROW\_DOFS, 1124  
 VECTOR, 1124

**TYPES::EQUATIONS\_COL\_TO\_SOLVER\_-  
   COLS\_MAP\_TYPE**, 1125  
 COUPLING\_COEFFICIENTS, 1125  
 NUMBER\_OF\_SOLVER\_COLS, 1125  
 SOLVER\_COLS, 1125

**TYPES::EQUATIONS\_INTERPOLATION\_-  
   TYPE**, 1126  
 DEPENDENT\_FIELD, 1127  
 DEPENDENT\_INTERP\_PARAMETERS,  
   1127  
 DEPENDENT\_INTERP\_POINT, 1127  
 EQUATIONS, 1127  
 FIBRE\_FIELD, 1128  
 FIBRE\_INTERP\_PARAMETERS, 1128  
 FIBRE\_INTERP\_POINT, 1128  
 FIBRE\_INTERP\_POINT\_METRICS, 1128  
 GEOMETRIC\_FIELD, 1128  
 GEOMETRIC\_INTERP\_PARAMETERS,  
   1128  
 GEOMETRIC\_INTERP\_POINT, 1128  
 GEOMETRIC\_INTERP\_POINT\_METRICS,  
   1129  
 MATERIAL\_FIELD, 1129  
 MATERIAL\_INTERP\_PARAMETERS, 1129  
 MATERIAL\_INTERP\_POINT, 1129  
 SOURCE\_FIELD, 1129  
 SOURCE\_INTERP\_PARAMETERS, 1129

SOURCE\_INTERP\_POINT, 1129  
TYPES::EQUATIONS\_JACOBIAN\_TO\_-  
    VARIABLE\_MAP\_TYPE, 1131  
COLUMN\_DOFS\_MAPPING, 1131  
EQUATIONS\_COLUMN\_TO\_DOF\_-  
    VARIABLE\_MAP, 1131  
JACOBIAN, 1131  
JACOBIAN\_COEFFICIENT, 1132  
NUMBER\_OF\_COLUMNS, 1132  
VARIABLE, 1132  
VARIABLE\_TYPE, 1132  
TYPES::EQUATIONS\_JACOBIAN\_TYPE, 1133  
ELEMENT\_JACOBIAN, 1133  
ELEMENT\_RESIDUAL, 1133  
JACOBIAN, 1134  
JACOBIAN\_CALCULATION\_TYPE, 1134  
NONLINEAR\_MATRICES, 1134  
NUMBER\_OF\_COLUMNS, 1134  
STORAGE\_TYPE, 1134  
STRUCTURE\_TYPE, 1134  
UPDATE\_JACOBIAN, 1134  
TYPES::EQUATIONS\_LINEAR\_DATA\_TYPE,  
    1136  
    EQUATIONS, 1136  
TYPES::EQUATIONS\_MAPPING\_CREATE\_-  
    VALUES\_CACHE\_TYPE, 1137  
MATRIX\_COEFFICIENTS, 1137  
MATRIX\_VARIABLE\_TYPES, 1137  
NUMBER\_OF\_LINEAR\_EQUATIONS\_-  
    MATRICES, 1138  
RESIDUAL\_COEFFICIENT, 1138  
RESIDUAL\_VARIABLE\_TYPE, 1138  
RHS\_COEFFICIENT, 1138  
RHS\_VARIABLE\_TYPE, 1138  
SOURCE\_COEFFICIENT, 1138  
SOURCE\_VARIABLE\_TYPE, 1138  
TYPES::EQUATIONS\_MAPPING\_LINEAR\_-  
    TYPE, 1140  
    EQUATIONS\_MAPPING, 1140  
    EQUATIONS\_MATRIX\_TO\_VARIABLE\_-  
        MAPS, 1140  
    EQUATIONS\_ROW\_TO\_VARIABLE\_-  
        DOF\_MAPS, 1141  
    MATRIX\_VARIABLE\_TYPES, 1141  
    NUMBER\_OF\_LINEAR\_EQUATIONS\_-  
        MATRICES, 1141  
    NUMBER\_OF\_LINEAR\_MATRIX\_-  
        VARIABLES, 1141  
    VARIABLE\_TO\_EQUATIONS\_-  
        MATRICES\_MAPS, 1141  
TYPES::EQUATIONS\_MAPPING\_-  
    NONLINEAR\_TYPE, 1142  
    EQUATIONS\_MAPPING, 1142  
EQUATIONS\_ROW\_TO\_SOURCE\_DOF\_-  
    MAP, 1147  
SOURCE\_COEFFICIENT, 1148  
SOURCE\_DOF\_TO\_EQUATIONS\_ROW\_-  
    MAP, 1148  
SOURCE\_VARIABLE, 1148  
SOURCE\_VARIABLE\_MAPPING, 1148  
SOURCE\_VARIABLE\_TYPE, 1148  
TYPES::EQUATIONS\_MAPPING\_TYPE, 1149  
CREATE\_VALUES\_CACHE, 1150  
EQUATIONS, 1150  
EQUATIONS\_MAPPING\_FINISHED, 1150  
EQUATIONS\_MATRICES, 1150  
LINEAR\_MAPPING, 1150  
NONLINEAR\_MAPPING, 1150  
NUMBER\_OF\_GLOBAL\_ROWS, 1150  
NUMBER\_OF\_ROWS, 1150  
RESIDUAL\_VARIABLE, 1150  
RESIDUAL\_VARIABLE\_MAPPING, 1151  
RESIDUAL\_VARIABLE\_TYPE, 1151  
RHS\_MAPPING, 1151  
ROW\_DOFS\_MAPPING, 1151  
SOURCE\_MAPPING, 1151  
TOTAL\_NUMBER\_OF\_ROWS, 1151  
TYPES::EQUATIONS\_MATRICES\_LINEAR\_-  
    TYPE, 1152  
    EQUATIONS\_MATRICES, 1152  
    MATRICES, 1152  
    NUMBER\_OF\_LINEAR\_MATRICES, 1152

TYPES::EQUATIONS\_MATRICES\_-  
     NONLINEAR\_TYPE, 1153  
     ELEMENT\_RESIDUAL, 1153  
     EQUATIONS\_MATRICES, 1153  
     JACOBIAN, 1153  
     RESIDUAL, 1153  
     UPDATE\_RESIDUAL, 1154  
 TYPES::EQUATIONS\_MATRICES\_RHS\_TYPE,  
     1155  
     ELEMENT\_VECTOR, 1155  
     EQUATIONS\_MATRICES, 1155  
     UPDATE\_VECTOR, 1155  
     VECTOR, 1155  
 TYPES::EQUATIONS\_MATRICES\_SOURCE\_-  
     TYPE, 1157  
     ELEMENT\_VECTOR, 1157  
     EQUATIONS\_MATRICES, 1157  
     UPDATE\_VECTOR, 1157  
     VECTOR, 1157  
 TYPES::EQUATIONS\_MATRICES\_TYPE, 1159  
     EQUATIONS, 1159  
     EQUATIONS\_MAPPING, 1159  
     EQUATIONS\_MATRICES\_FINISHED, 1160  
     LINEAR\_MATRICES, 1160  
     NONLINEAR\_MATRICES, 1160  
     NUMBER\_OF\_GLOBAL\_ROWS, 1160  
     NUMBER\_OF\_ROWS, 1160  
     RHS\_VECTOR, 1160  
     SOLUTION\_MAPPING, 1160  
     SOURCE\_VECTOR, 1160  
     TOTAL\_NUMBER\_OF\_ROWS, 1161  
 TYPES::EQUATIONS\_MATRIX\_PTR\_TYPE,  
     1162  
     PTR, 1162  
 TYPES::EQUATIONS\_MATRIX\_TO\_-  
     VARIABLE\_MAP\_TYPE, 1163  
     COLUMN\_DOFS\_MAPPING, 1163  
     COLUMN\_TO\_DOF\_MAP, 1163  
     EQUATIONS\_MATRIX, 1164  
     MATRIX\_COEFFICIENT, 1164  
     MATRIX\_NUMBER, 1164  
     NUMBER\_OF\_COLUMNS, 1164  
     VARIABLE, 1164  
     VARIABLE\_TYPE, 1164  
 TYPES::EQUATIONS\_MATRIX\_TYPE, 1165  
     ELEMENT\_MATRIX, 1165  
     LINEAR\_MATRICES, 1165  
     MATRIX, 1166  
     MATRIX\_NUMBER, 1166  
     NUMBER\_OF\_COLUMNS, 1166  
     STORAGE\_TYPE, 1166  
     STRUCTURE\_TYPE, 1166  
     UPDATE\_MATRIX, 1166  
 TYPES::EQUATIONS\_NONLINEAR\_DATA\_-  
     TYPE, 1167  
     EQUATIONS, 1167  
 TYPES::EQUATIONS\_ROW\_TO\_SOLVER\_-  
     ROWS\_MAP\_TYPE, 1168  
     COUPLING\_COEFFICIENTS, 1168  
     NUMBER\_OF\_SOLVER\_ROWS, 1168  
     SOLVER\_ROWS, 1168  
 TYPES::EQUATIONS\_SET\_ANALYTIC\_TYPE,  
     1169  
     ANALYTIC\_FINISHED, 1169  
     EQUATION\_NUMBER, 1169  
     EQUATIONS\_SET, 1169  
 TYPES::EQUATIONS\_SET\_DEPENDENT\_-  
     TYPE, 1170  
     DEPENDENT\_FIELD, 1170  
     DEPENDENT\_FINISHED, 1170  
     EQUATIONS\_SET, 1170  
 TYPES::EQUATIONS\_SET\_FIXED\_-  
     CONDITIONS\_TYPE, 1171  
     BOUNDARY\_CONDITIONS, 1171  
     EQUATIONS\_SET, 1171  
     FIXED\_CONDITIONS\_FINISHED, 1171  
     GLOBAL\_BOUNDARY\_CONDITIONS,  
         1171  
 TYPES::EQUATIONS\_SET\_GEOMETRY\_TYPE,  
     1173  
     EQUATIONS\_SET, 1173  
     FIBRE\_FIELD, 1173  
     GEOMETRIC\_FIELD, 1173  
 TYPES::EQUATIONS\_SET\_MATERIALS\_-  
     TYPE, 1174  
     EQUATIONS\_SET, 1174  
     MATERIAL\_FIELD, 1174  
     MATERIALS\_FINISHED, 1174  
 TYPES::EQUATIONS\_SET\_PTR\_TYPE, 1175  
     PTR, 1175  
 TYPES::EQUATIONS\_SET\_SOURCE\_TYPE,  
     1176  
     EQUATIONS\_SET, 1176  
     SOURCE\_FIELD, 1176  
     SOURCE\_FINISHED, 1176  
 TYPES::EQUATIONS\_SET\_TO\_SOLVER\_-  
     MAP\_TYPE, 1177  
     EQUATIONS, 1177  
     EQUATIONS\_ROW\_TO\_SOLVER\_ROWS\_-  
         MAPS, 1177  
     EQUATIONS\_SET\_INDEX, 1178  
     EQUATIONS\_TO\_SOLVER\_MATRIX\_-  
         MAPS\_EM, 1178  
     EQUATIONS\_TO\_SOLVER\_MATRIX\_-  
         MAPS\_JM, 1178  
     EQUATIONS\_TO\_SOLVER\_MATRIX\_-  
         MAPS\_SM, 1178

SOLUTION\_MAPPING, 1178  
TYPES::EQUATIONS\_SET\_TYPE, 1179  
ANALYTIC, 1180  
CLASS, 1180  
DEPENDENT, 1180  
EQUATIONS, 1180  
EQUATIONS\_SET\_FINISHED, 1180  
EQUATIONS\_SETS, 1180  
FIXED\_CONDITIONS, 1181  
GEOMETRY, 1181  
GLOBAL\_NUMBER, 1181  
LINEARITY, 1181  
MATERIALS, 1181  
REGION, 1181  
SOLUTION\_METHOD, 1181  
SOURCE, 1182  
SUBTYPE, 1182  
TIME\_TYPE, 1182  
TYPE, 1182  
USER\_NUMBER, 1182  
TYPES::EQUATIONS\_SETS\_TYPE, 1183  
EQUATIONS\_SETS, 1183  
NUMBER\_OF\_EQUATIONS\_SETS, 1183  
REGION, 1183  
TYPES::EQUATIONS\_TIME\_DATA\_TYPE, 1184  
EQUATIONS, 1184  
TYPES::EQUATIONS\_TO\_SOLVER\_MAPS\_-  
PTR\_TYPE, 1185  
PTR, 1185  
TYPES::EQUATIONS\_TO\_SOLVER\_MAPS\_-  
TYPE, 1186  
EQUATIONS\_COL\_SOLVER\_COLS\_MAP,  
1186  
EQUATIONS\_MATRIX, 1186  
EQUATIONS\_MATRIX\_NUMBER, 1186  
SOLVER\_MATRIX, 1187  
SOLVER\_MATRIX\_NUMBER, 1187  
TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_-  
MAPS\_EM\_TYPE, 1188  
EQUATIONS\_MATRIX\_NUMBER, 1188  
EQUATIONS\_TO\_SOLVER\_MATRIX\_-  
MAPS, 1188  
NUMBER\_OF\_SOLVER\_MATRICES, 1188  
TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_-  
MAPS\_JM\_TYPE, 1190  
JACOBIAN\_TO\_SOLVER\_MATRIX\_MAP,  
1190  
TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_-  
MAPS\_SM\_TYPE, 1191  
EQUATIONS\_TO\_SOLVER\_MATRIX\_-  
MAPS, 1192  
JACOBIAN\_TO\_SOLVER\_MATRIX\_MAP,  
1192  
NUMBER\_OF\_LINEAR\_EQUATIONS\_-  
MATRICES, 1192  
NUMBER\_OF\_VARIABLES, 1192  
SOLVER\_MATRIX\_NUMBER, 1192  
VARIABLE\_TO\_SOLVER\_COL\_MAPS,  
1192  
VARIABLE\_TYPES, 1192  
VARIABLES, 1193  
TYPES::EQUATIONS\_TYPE, 1194  
EQUATIONS\_FINISHED, 1195  
EQUATIONS\_MAPPING, 1195  
EQUATIONS\_MATRICES, 1195  
EQUATIONS\_SET, 1195  
INTERPOLATION, 1195  
LINEAR\_DATA, 1195  
NONLINEAR\_DATA, 1195  
NONLINEAR\_JACOBIAN\_TYPE, 1195  
OUTPUT\_TYPE, 1196  
SPARSITY\_TYPE, 1196  
TIME\_DATA, 1196  
TYPES::FIELD\_CREATE\_VALUES\_CACHE\_-  
TYPE, 1197  
INTERPOLATION\_TYPE, 1197  
MESH\_COMPONENT\_NUMBER, 1197  
NUMBER\_OF\_COMPONENTS, 1197  
VARIABLE\_TYPES, 1198  
TYPES::FIELD\_DOF\_TO\_PARAM\_MAP\_TYPE,  
1199  
CONSTANT\_DOF2PARAM\_MAP, 1200  
DOF\_TYPE, 1200  
ELEMENT\_DOF2PARAM\_MAP, 1200  
NODE\_DOF2PARAM\_MAP, 1200  
NUMBER\_OF\_CONSTANT\_DOFS, 1201  
NUMBER\_OF\_DOFS, 1201  
NUMBER\_OF\_ELEMENT\_DOFS, 1201  
NUMBER\_OF\_NODE\_DOFS, 1201  
NUMBER\_OF\_POINT\_DOFS, 1201  
POINT\_DOF2PARAM\_MAP, 1201  
VARIABLE\_DOF, 1202  
TYPES::FIELD\_GEOMETRIC\_PARAMETERS\_-  
TYPE, 1203  
AREAS, 1203  
FIELDS\_USING, 1203  
LENGTHS, 1204  
NUMBER\_OF AREAS, 1204  
NUMBER\_OF\_FIELDS\_USING, 1204  
NUMBER\_OF\_LINES, 1204  
NUMBER\_OF\_VOLUMES, 1204  
VOLUMES, 1204  
TYPES::FIELD\_INTERPOLATED\_POINT\_-  
METRICS\_TYPE, 1205  
DX\_DXI, 1205  
DXI\_DX, 1205  
GL, 1206

GU, 1206  
 INTERPOLATED\_POINT, 1206  
 JACOBIAN, 1206  
 JACOBIAN\_TYPE, 1206  
 NUMBER\_OF\_X\_DIMENSIONS, 1206  
 NUMBER\_OF\_XI\_DIMENSIONS, 1206  
 TYPES::FIELD\_INTERPOLATED\_POINT\_-  
     TYPE, 1208  
     INTERPOLATION\_PARAMETERS, 1208  
     MAX\_PARTIAL\_DERIVATIVE\_INDEX,  
         1208  
     PARTIAL\_DERIVATIVE\_TYPE, 1208  
     VALUES, 1209  
 TYPES::FIELD\_INTERPOLATION\_-  
     PARAMETERS\_TYPE, 1210  
     BASES, 1210  
     FIELD, 1210  
     FIELD\_VARIABLE, 1210  
     NUMBER\_OF\_PARAMETERS, 1211  
     PARAMETERS, 1211  
 TYPES::FIELD\_MAPPINGS\_TYPE, 1212  
     DOF\_TO\_PARAM\_MAP, 1212  
     DOMAIN\_MAPPING, 1212  
 TYPES::FIELD\_PARAM\_TO\_DOF\_MAP\_TYPE,  
     1213  
     CONSTANT\_PARAM2DOF\_MAP, 1214  
     ELEMENT\_PARAM2DOF\_MAP, 1214  
     MAX\_NUMBER\_OF\_DERIVATIVES, 1214  
     NODE\_PARAM2DOF\_MAP, 1214  
     NUMBER\_OF\_CONSTANT\_-  
         PARAMETERS, 1214  
     NUMBER\_OF\_ELEMENT\_PARAMETERS,  
         1214  
     NUMBER\_OF\_NODE\_PARAMETERS,  
         1215  
     NUMBER\_OF\_POINT\_PARAMETERS,  
         1215  
     POINT\_PARAM2DOF\_MAP, 1215  
 TYPES::FIELD\_PARAMETER\_SET\_PTR\_-  
     TYPE, 1216  
     PTR, 1216  
 TYPES::FIELD\_PARAMETER\_SET\_TYPE, 1217  
     PARAMETERS, 1217  
     SET\_INDEX, 1217  
     SET\_TYPE, 1217  
 TYPES::FIELD\_PARAMETER\_SETS\_TYPE,  
     1219  
     FIELD, 1219  
     NUMBER\_OF\_PARAMETER\_SETS, 1219  
     PARAMETER\_SETS, 1219  
     SET\_TYPE, 1220  
 TYPES::FIELD\_PTR\_TYPE, 1221  
     PTR, 1221  
 TYPES::FIELD\_SCALING\_TYPE, 1222  
     MAX\_NUMBER\_OF\_DERIVATIVES, 1222  
     MAX\_NUMBER\_OF\_ELEMENT\_-  
         PARAMETERS, 1222  
     MESH\_COMPONENT\_NUMBER, 1222  
     SCALE\_FACTORS, 1222  
 TYPES::FIELD\_SCALINGS\_TYPE, 1224  
     NUMBER\_OF\_SCALING\_INDICES, 1224  
     SCALING\_TYPE, 1224  
     SCALINGS, 1224  
 TYPES::FIELD\_TYPE, 1225  
     DECOMPOSITION, 1226  
     DEPENDENT\_TYPE, 1226  
     DIMENSION, 1226  
     FIELD\_FINISHED, 1226  
     FIELDS, 1227  
     GEOMETRIC\_FIELD, 1227  
     GEOMETRIC\_FIELD\_PARAMETERS, 1227  
     GLOBAL\_NUMBER, 1227  
     MAPPINGS, 1227  
     NUMBER\_OF\_VARIABLES, 1227  
     PARAMETER\_SETS, 1227  
     REGION, 1227  
     SCALINGS, 1228  
     TYPE, 1228  
     USER\_NUMBER, 1228  
     VARIABLE\_TYPE\_MAP, 1228  
     VARIABLES, 1228  
 TYPES::FIELD\_VARIABLE\_COMPONENT\_-  
     TYPE, 1229  
     COMPONENT\_NUMBER, 1230  
     DOMAIN, 1230  
     FIELD, 1230  
     FIELD\_VARIABLE, 1230  
     INTERPOLATION\_TYPE, 1230  
     MAX\_NUMBER\_OF\_INTERPOLATION\_-  
         PARAMETERS, 1230  
     MESH\_COMPONENT\_NUMBER, 1230  
     PARAM\_TO\_DOF\_MAP, 1230  
     REGION, 1231  
     SCALING\_INDEX, 1231  
 TYPES::FIELD\_VARIABLE\_PTR\_TYPE, 1232  
     PTR, 1232  
 TYPES::FIELD\_VARIABLE\_TYPE, 1233  
     COMPONENTS, 1234  
     DOF\_LIST, 1234  
     DOMAIN\_MAPPING, 1234  
     FIELD, 1234  
     GLOBAL\_DOF\_OFFSET, 1234  
     MAX\_NUMBER\_OF\_INTERPOLATION\_-  
         PARAMETERS, 1234  
     NUMBER\_OF\_COMPONENTS, 1234  
     NUMBER\_OF\_DOFS, 1235  
     NUMBER\_OF\_GLOBAL\_DOFS, 1235  
     REGION, 1235

TOTAL\_NUMBER\_OF\_DOFS, 1235  
VARIABLE\_NUMBER, 1235  
VARIABLE\_TYPE, 1235  
TYPES::FIELDS\_TYPE, 1236  
FIELDS, 1236  
NUMBER\_OF\_FIELDS, 1236  
REGION, 1236  
TYPES::GENERATED\_MESH\_PTR\_TYPE, 1237  
PTR, 1237  
TYPES::GENERATED\_MESH\_REGULAR\_TYPE, 1238  
BASIS, 1238  
GENERATED\_MESH, 1238  
MAXIMUM\_EXTENT, 1238  
MESH\_DIMENSION, 1239  
NUMBER\_OF\_ELEMENTS\_XI, 1239  
ORIGIN, 1239  
TYPES::GENERATED\_MESH\_TYPE, 1240  
GENERATED\_MESH\_FINISHED, 1240  
GENERATED\_TYPE, 1240  
GLOBAL\_NUMBER, 1240  
MESH, 1240  
REGION, 1240  
REGULAR\_MESH, 1240  
USER\_NUMBER, 1241  
TYPES::GENERATED\_MESHS\_TYPE, 1242  
GENERATED\_MESHS, 1242  
NUMBER\_OF\_GENERATED\_MESHS, 1242  
TYPES::JACOBIAN\_COL\_TO\_SOLVER\_COLS\_MAP\_TYPE, 1243  
COUPLING\_COEFFICIENTS, 1243  
NUMBER\_OF\_SOLVER\_COLS, 1243  
SOLVER\_COLS, 1243  
TYPES::JACOBIAN\_TO\_SOLVER\_MAP\_TYPE, 1244  
JACOBIAN\_COL\_SOLVER\_COLS\_MAP, 1244  
JACOBIAN\_MATRIX, 1244  
SOLVER\_MATRIX, 1244  
SOLVER\_MATRIX\_NUMBER, 1244  
TYPES::LINEAR\_DIRECT\_SOLVER\_TYPE, 1246  
DIRECT\_SOLVER\_TYPE, 1246  
LINEAR\_SOLVER, 1246  
SOLVER\_LIBRARY, 1246  
TYPES::LINEAR\_ITERATIVE\_SOLVER\_TYPE, 1247  
ABSOLUTE\_TOLERANCE, 1248  
DIVERGENCE\_TOLERANCE, 1248  
ITERATIVE\_PRECONDITIONER\_TYPE, 1248  
ITERATIVE\_SOLVER\_TYPE, 1248  
KSP, 1248  
LINEAR\_SOLVER, 1248  
MAXIMUM\_NUMBER\_OF\_ITERATIONS, 1248  
PC, 1248  
RELATIVE\_TOLERANCE, 1249  
SOLVER\_LIBRARY, 1249  
TYPES::LINEAR\_SOLVER\_TYPE, 1250  
DIRECT\_SOLVER, 1250  
ITERATIVE\_SOLVER, 1250  
LINEAR\_SOLVE\_TYPE, 1250  
SOLVER, 1250  
TYPES::MATRIX\_TYPE, 1252  
COLUMN\_INDICES, 1253  
DATA\_DP, 1253  
DATA\_INTG, 1253  
DATA\_L, 1253  
DATA\_SP, 1253  
DATA\_TYPE, 1254  
ID, 1254  
M, 1254  
MATRIX\_FINISHED, 1254  
MAX\_M, 1254  
MAX\_N, 1254  
MAXIMUM\_COLUMN\_INDICES\_PER\_ROW, 1254  
N, 1254  
NUMBER\_NON\_ZEROS, 1255  
ROW\_INDICES, 1255  
SIZE, 1255  
STORAGE\_TYPE, 1255  
TYPES::MESH\_DOFS\_TYPE, 1256  
MESH, 1256  
NUMBER\_OF\_DOFS, 1256  
TYPES::MESH\_ELEMENT\_TYPE, 1257  
ADJACENT\_ELEMENTS, 1257  
BASIS, 1258  
GLOBAL\_ELEMENT\_NODES, 1258  
GLOBAL\_NUMBER, 1258  
NUMBER\_OF\_ADJACENT\_ELEMENTS, 1258  
USER\_ELEMENT\_NODES, 1258  
USER\_NUMBER, 1258  
TYPES::MESH\_ELEMENTS\_TYPE, 1259  
ELEMENTS, 1259  
ELEMENTS\_FINISHED, 1259  
MESH, 1259  
NUMBER\_OF\_ELEMENTS, 1260  
TYPES::MESH\_NODE\_TYPE, 1261  
DOF\_INDEX, 1261  
GLOBAL\_NUMBER, 1261  
NUMBER\_OF\_DERIVATIVES, 1261  
NUMBER\_OF\_SURROUNDING\_ELEMENTS, 1262  
PARTIAL\_DERIVATIVE\_INDEX, 1262

SURROUNDING\_ELEMENTS, 1262  
 USER\_NUMBER, 1262  
**TYPES::MESH\_NODES\_TYPE, 1263**  
   MESH, 1263  
   NODES, 1263  
   NUMBER\_OF\_NODES, 1263  
**TYPES::MESH\_PTR\_TYPE, 1264**  
   PTR, 1264  
**TYPES::MESH\_TOPOLOGY\_PTR\_TYPE, 1265**  
   PTR, 1265  
**TYPES::MESH\_TOPOLOGY\_TYPE, 1266**  
   DOFS, 1266  
   ELEMENTS, 1266  
   MESH, 1266  
   MESH\_COMPONENT\_NUMBER, 1266  
   NODES, 1267  
**TYPES::MESH\_TYPE, 1268**  
   DECOMPOSITIONS, 1269  
   EMBEDDED\_MESHES, 1269  
   EMBEDDING\_MESH, 1269  
   GENERATED\_MESH, 1269  
   GLOBAL\_NUMBER, 1269  
   MESH\_EMBEDDED, 1269  
   MESH\_FINISHED, 1270  
   MESHES, 1270  
   NUMBER\_OF\_COMPONENTS, 1270  
   NUMBER\_OF\_DIMENSIONS, 1270  
   NUMBER\_OF\_ELEMENTS, 1270  
   NUMBER\_OF\_EMBEDDED\_MESHES,  
     1270  
   NUMBER\_OF\_FACES, 1270  
   NUMBER\_OF\_LINES, 1270  
   REGION, 1270  
   TOPOLOGY, 1271  
   USER\_NUMBER, 1271  
**TYPES::MESHES\_TYPE, 1272**  
   MESHES, 1272  
   NUMBER\_OF\_MESHES, 1272  
   REGION, 1272  
**TYPES::NODE\_TYPE, 1273**  
   GLOBAL\_NUMBER, 1273  
   INITIAL\_POSITION, 1273  
   LABEL, 1273  
   USER\_NUMBER, 1273  
**TYPES::NODES\_TYPE, 1275**  
   NODE\_TREE, 1275  
   NODES, 1275  
   NODES\_FINISHED, 1275  
   NUMBER\_OF\_NODES, 1276  
   REGION, 1276  
**TYPES::NONLINEAR\_LINESEARCH\_-  
   SOLVER\_TYPE, 1277**  
   JACOBIAN\_CALCULATION\_TYPE, 1278  
   JACOBIAN\_FDCOLORING, 1278  
   JACOBIAN\_ISCOLORING, 1278  
   LINESEARCH\_ALPHA, 1278  
   LINESEARCH\_MAXSTEP, 1278  
   LINESEARCH\_STEPTOLERANCE, 1278  
   LINESEARCH\_TYPE, 1278  
   NONLINEAR\_SOLVER, 1279  
   SNES, 1279  
   SOLVER\_LIBRARY, 1279  
**TYPES::NONLINEAR\_SOLVER\_TYPE, 1280**  
   ABSOLUTE\_TOLERANCE, 1280  
   LINESEARCH\_SOLVER, 1280  
   MAXIMUM\_NUMBER\_OF\_FUNCTION\_-  
     EVALUATIONS, 1281  
   MAXIMUM\_NUMBER\_OF\_ITERATIONS,  
     1281  
   NONLINEAR\_SOLVE\_TYPE, 1281  
   RELATIVE\_TOLERANCE, 1281  
   SOLUTION\_TOLERANCE, 1281  
   SOLVER, 1281  
   TRUSTREGION\_SOLVER, 1281  
**TYPES::NONLINEAR\_TRUSTREGION\_-  
   SOLVER\_TYPE, 1283**  
   NONLINEAR\_SOLVER, 1283  
   SNES, 1283  
   SOLVER\_LIBRARY, 1283  
   TRUSTREGION\_DELTA0, 1284  
   TRUSTREGION\_TOLERANCE, 1284  
**TYPES::PROBLEM\_CONTROL\_TYPE, 1285**  
   CONTROL\_FINISHED, 1285  
   CONTROL\_TYPE, 1285  
   PROBLEM, 1285  
**TYPES::PROBLEM\_LINEAR\_DATA\_TYPE,  
   1286**  
   SOLUTION, 1286  
**TYPES::PROBLEM\_NONLINEAR\_DATA\_-  
   TYPE, 1287**  
   NUMBER\_OF\_ITERATIONS, 1287  
   SOLUTION, 1287  
**TYPES::PROBLEM\_PTR\_TYPE, 1288**  
   PTR, 1288  
**TYPES::PROBLEM\_TIME\_DATA\_TYPE, 1289**  
   SOLUTION, 1289  
**TYPES::PROBLEM\_TYPE, 1290**  
   CLASS, 1290  
   CONTROL, 1290  
   GLOBAL\_NUMBER, 1291  
   NUMBER\_OF\_SOLUTIONS, 1291  
   PROBLEM\_FINISHED, 1291  
   PROBLEMS, 1291  
   SOLUTIONS, 1291  
   SUBTYPE, 1291  
   TYPE, 1291  
   USER\_NUMBER, 1291  
**TYPES::PROBLEMS\_TYPE, 1293**

NUMBER\_OF\_PROBLEMS, 1293  
PROBLEMS, 1293  
TYPES::QUADRATURE\_SCHEME\_PTR\_TYPE, 1294  
PTR, 1294  
TYPES::QUADRATURE\_SCHEME\_TYPE, 1295  
GAUSS\_BASIS\_FNS, 1295  
GAUSS\_POSITIONS, 1295  
GAUSS\_WEIGHTS, 1296  
GLOBAL\_NUMBER, 1296  
NUMBER\_OF\_GAUSS, 1296  
QUADRATURE, 1296  
TYPES::QUADRATURE\_TYPE, 1297  
BASIS, 1297  
GAUSS\_ORDER, 1297  
NUMBER\_OF\_GAUSS\_XI, 1298  
NUMBER\_OF\_SCHEMES, 1298  
QUADRATURE\_SCHEME\_MAP, 1298  
SCHEMES, 1298  
TYPE, 1298  
TYPES::REGION\_PTR\_TYPE, 1299  
PTR, 1299  
TYPES::REGION\_TYPE, 1300  
COORDINATE\_SYSTEM, 1301  
EQUATIONS\_SETS, 1301  
FIELDS, 1301  
LABEL, 1301  
MESHES, 1301  
NODES, 1301  
NUMBER\_OF\_SUB\_REGIONS, 1301  
PARENT\_REGION, 1301  
REGION\_FINISHED, 1301  
SUB\_REGIONS, 1302  
USER\_NUMBER, 1302  
TYPES::SOLUTION\_MAPPING\_CREATE\_VALUES\_CACHE\_TYPE, 1303  
MATRIX\_VARIABLE\_TYPES, 1303  
RESIDUAL\_VARIABLE\_TYPE, 1303  
RHS\_VARIABLE\_TYPE, 1304  
SOURCE\_VARIABLE\_TYPE, 1304  
TYPES::SOLUTION\_MAPPING\_TYPE, 1305  
CREATE\_VALUES\_CACHE, 1306  
EQUATIONS\_SET\_TO\_SOLVER\_MAP, 1306  
EQUATIONS\_SETS, 1306  
HAVE\_JACOBIAN, 1306  
NUMBER\_OF\_EQUATIONS\_SETS, 1306  
NUMBER\_OF\_GLOBAL\_ROWS, 1306  
NUMBER\_OF\_ROWS, 1307  
NUMBER\_OF\_SOLVER\_MATRICES, 1307  
ROW\_DOFS\_MAPPING, 1307  
SOLUTION, 1307  
SOLUTION\_MAPPING\_FINISHED, 1307  
SOLVER\_COL\_TO\_EQUATIONS\_SETS\_MAP, 1307  
SOLVER\_ROW\_TO\_EQUATIONS\_SET\_MAPS, 1307  
TYPES::SOLUTION\_PTR\_TYPE, 1309  
PTR, 1309  
TYPES::SOLUTION\_TYPE, 1310  
EQUATIONS\_SET\_ADDED\_INDEX, 1310  
EQUATIONS\_SET\_TO\_ADD, 1310  
LINEARITY, 1310  
PROBLEM, 1311  
SOLUTION\_FINISHED, 1311  
SOLUTION\_MAPPING, 1311  
SOLUTION\_NUMBER, 1311  
SOLVER, 1311  
TYPES::SOLVER\_COL\_TO\_EQUATIONS\_MAP\_TYPE, 1312  
COUPLING\_COEFFICIENTS, 1312  
EQUATIONS\_COL\_NUMBERS, 1312  
EQUATIONS\_MATRIX\_NUMBERS, 1312  
JACOBIAN\_COL\_NUMBER, 1313  
JACOBIAN\_COUPLING\_COEFFICIENT, 1313  
NUMBER\_OF\_LINEAR\_EQUATIONS\_MATRICES, 1313  
TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SET\_MAP\_TYPE, 1314  
EQUATIONS, 1314  
SOLVER\_COL\_TO\_EQUATIONS\_MAPS, 1314  
TYPES::SOLVER\_COL\_TO\_EQUATIONS\_SETS\_MAP\_TYPE, 1315  
COLUMN\_DOFS\_MAPPING, 1316  
NUMBER\_OF\_COLUMNS, 1316  
NUMBER\_OF\_DOFS, 1316  
NUMBER\_OF\_GLOBAL\_DOFs, 1316  
SOLUTION\_MAPPING, 1316  
SOLVER\_COL\_TO\_EQUATIONS\_SET\_MAPS, 1316  
SOLVER\_DOF\_TO\_VARIABLE\_MAPS, 1316  
SOLVER\_MATRIX, 1316  
SOLVER\_MATRIX\_NUMBER, 1317  
TOTAL\_NUMBER\_OF\_DOFs, 1317  
TYPES::SOLVER\_DOF\_TO\_VARIABLE\_MAP\_TYPE, 1318  
ADDITIVE\_CONSTANT, 1318  
EQUATIONS\_SET\_INDICES, 1318  
NUMBER\_OF\_EQUATIONS\_SETS, 1318  
VARIABLE, 1319  
VARIABLE\_COEFFICIENT, 1319  
VARIABLE\_DOF, 1319  
TYPES::SOLVER\_JACOBIAN\_TYPE, 1320  
JACOBIAN, 1320

NUMBER\_OF\_COLUMNS, 1320  
 SOLVER\_MATRICES, 1320  
 STORAGE\_TYPE, 1320  
 UPDATE\_JACOBIAN, 1321  
 TYPES::SOLVER\_MATRICES\_TYPE, 1322  
 LIBRARY\_TYPE, 1323  
 MATRICES, 1323  
 NUMBER\_OF\_GLOBAL\_ROWS, 1323  
 NUMBER\_OF\_MATRICES, 1323  
 NUMBER\_OF\_ROWS, 1323  
 RESIDUAL, 1323  
 RHS\_VECTOR, 1323  
 SOLUTION\_MAPPING, 1323  
 SOLVER, 1324  
 SOLVER\_MATRICES\_FINISHED, 1324  
 UPDATE\_RESIDUAL, 1324  
 UPDATE\_RHS\_VECTOR, 1324  
 TYPES::SOLVER\_MATRIX\_PTR\_TYPE, 1325  
 PTR, 1325  
 TYPES::SOLVER\_MATRIX\_TYPE, 1326  
 MATRIX, 1326  
 MATRIX\_NUMBER, 1326  
 NUMBER\_OF\_COLUMNS, 1326  
 SOLVER\_MATRICES, 1327  
 SOLVER\_VECTOR, 1327  
 STORAGE\_TYPE, 1327  
 UPDATE\_MATRIX, 1327  
 TYPES::SOLVER\_ROW\_TO\_EQUATIONS\_SET\_MAP\_TYPE, 1328  
 COUPLING\_COEFFICIENTS, 1328  
 EQUATIONS\_ROW\_NUMBER, 1328  
 EQUATIONS\_SET, 1328  
 NUMBER\_OF\_ROWS, 1328  
 TYPES::SOLVER\_TYPE, 1330  
 EIGENPROBLEM\_SOLVER, 1331  
 LINEAR\_SOLVER, 1331  
 NONLINEAR\_SOLVER, 1331  
 OUTPUT\_TYPE, 1331  
 SOLUTION, 1331  
 SOLUTION\_MAPPING, 1331  
 SOLVE\_TYPE, 1331  
 SOLVER\_FINISHED, 1332  
 SOLVER\_MATRICES, 1332  
 SPARSITY\_TYPE, 1332  
 TIME\_INTEGRATION\_SOLVER, 1332  
 TYPES::TIME\_INTEGRATION\_SOLVER\_TYPE, 1333  
 SOLVER, 1333  
 SOLVER\_LIBRARY, 1333  
 TYPES::VARIABLE\_TO\_EQUATIONS\_COLUMN\_MAP\_TYPE, 1334  
 COLUMN\_DOF, 1334  
 TYPES::VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_MAP\_TYPE, 1335  
 DOF\_TO\_COLUMNS\_MAP, 1335  
 DOF\_TO\_ROWS\_MAP, 1335  
 VARIABLE, 1335  
 VARIABLE\_TYPE, 1335  
 TYPES::VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP\_TYPE, 1337  
 DOF\_TO\_COLUMNS\_MAPS, 1337  
 DOF\_TO\_ROWS\_MAP, 1337  
 EQUATIONS\_MATRIX\_NUMBERS, 1338  
 NUMBER\_OF\_LINEAR\_EQUATIONS\_MATRICES, 1338  
 VARIABLE, 1338  
 VARIABLE\_INDEX, 1338  
 VARIABLE\_TYPE, 1338  
 TYPES::VARIABLE\_TO\_SOLVER\_COL\_MAP\_TYPE, 1339  
 ADDITIVE\_CONSTANTS, 1339  
 COLUMN\_NUMBERS, 1339  
 COUPLING\_COEFFICIENTS, 1339  
 TYPES::VECTOR\_TYPE, 1341  
 DATA\_DP, 1341  
 DATA\_INTG, 1341  
 DATA\_L, 1342  
 DATA\_SP, 1342  
 DATA\_TYPE, 1342  
 ID, 1342  
 N, 1342  
 SIZE, 1342  
 VECTOR\_FINISHED, 1342  
 UNPACKCHARACTERS  
 F90C::interface, 844  
 UnPackCharacters  
 f90c\_c.c, 1412  
 UPDATE\_JACOBIAN  
 TYPES::EQUATIONS\_JACOBIAN\_TYPE, 1134  
 TYPES::SOLVER\_JACOBIAN\_TYPE, 1321  
 UPDATE\_MATRIX  
 TYPES::EQUATIONS\_MATRIX\_TYPE, 1166  
 TYPES::SOLVER\_MATRIX\_TYPE, 1327  
 UPDATE\_RESIDUAL  
 TYPES::EQUATIONS\_MATRICES\_NONLINEAR\_TYPE, 1154  
 TYPES::SOLVER\_MATRICES\_TYPE, 1324  
 UPDATE\_RHS\_VECTOR  
 TYPES::SOLVER\_MATRICES\_TYPE, 1324  
 UPDATE\_VECTOR  
 TYPES::EQUATIONS\_MATRICES\_RHS\_TYPE, 1155  
 TYPES::EQUATIONS\_MATRICES\_SOURCE\_TYPE, 1157  
 USER\_CPU

TIMER, 746  
timer\_c.c, 1534

USER\_ELEMENT\_NODES  
  TYPES::MESH\_ELEMENT\_TYPE, 1258

USER\_NUMBER  
  TYPES::BASIS\_TYPE, 1051  
  TYPES::COORDINATE\_SYSTEM\_TYPE, 1054  
  TYPES::DECOMPOSITION\_ELEMENT\_TYPE, 1056  
  TYPES::DECOMPOSITION\_TYPE, 1066  
  TYPES::DOMAIN\_NODE\_TYPE, 1113  
  TYPES::EQUATIONS\_SET\_TYPE, 1182  
  TYPES::FIELD\_TYPE, 1228  
  TYPES::GENERATED\_MESH\_TYPE, 1241  
  TYPES::MESH\_ELEMENT\_TYPE, 1258  
  TYPES::MESH\_NODE\_TYPE, 1262  
  TYPES::MESH\_TYPE, 1271  
  TYPES::NODE\_TYPE, 1273  
  TYPES::PROBLEM\_TYPE, 1291  
  TYPES::REGION\_TYPE, 1302

VALUE  
  TREES::TREE\_NODE\_TYPE, 1037

VALUES  
  TYPES::FIELD\_INTERPOLATED\_POINT\_TYPE, 1209

var\_str\_  
  ISO\_VARYING\_STRING, 511  
  ISO\_VARYING\_STRING::var\_str, 935

VARIABLE  
  TYPES::EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP\_TYPE, 1132  
  TYPES::EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_TYPE, 1164  
  TYPES::SOLVER\_DOF\_TO\_VARIABLE\_MAP\_TYPE, 1319  
  TYPES::VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_MAP\_TYPE, 1335  
  TYPES::VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP\_TYPE, 1338

VARIABLE\_COEFFICIENT  
  TYPES::SOLVER\_DOF\_TO\_VARIABLE\_MAP\_TYPE, 1319

VARIABLE\_DOF  
  TYPES::FIELD\_DOF\_TO\_PARAM\_MAP\_TYPE, 1202  
  TYPES::SOLVER\_DOF\_TO\_VARIABLE\_MAP\_TYPE, 1319

VARIABLE\_INDEX  
  TYPES::VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP\_TYPE, 1338

VARIABLE\_NUMBER  
  TYPES::FIELD\_VARIABLE\_TYPE, 1235

VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAPS  
  TYPES::EQUATIONS\_MAPPING\_LINEAR\_TYPE, 1141

VARIABLE\_TO\_JACOBIAN\_MAP  
  TYPES::EQUATIONS\_MAPPING\_NONLINEAR\_TYPE, 1143

VARIABLE\_TO\_SOLVER\_COL\_MAPS  
  TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_SM\_TYPE, 1192

VARIABLE\_TYPE  
  TYPES::EQUATIONS\_JACOBIAN\_TO\_VARIABLE\_MAP\_TYPE, 1132  
  TYPES::EQUATIONS\_MATRIX\_TO\_VARIABLE\_MAP\_TYPE, 1164  
  TYPES::FIELD\_VARIABLE\_TYPE, 1235  
  TYPES::VARIABLE\_TO\_EQUATIONS\_JACOBIAN\_MAP\_TYPE, 1335  
  TYPES::VARIABLE\_TO\_EQUATIONS\_MATRICES\_MAP\_TYPE, 1338

VARIABLE\_TYPE\_MAP  
  TYPES::FIELD\_TYPE, 1228

VARIABLE\_TYPES  
  TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_SM\_TYPE, 1192  
  TYPES::FIELD\_CREATE\_VALUES\_CACHE\_TYPE, 1198

VARIABLES  
  TYPES::EQUATIONS\_TO\_SOLVER\_MATRIX\_MAPS\_SM\_TYPE, 1193  
  TYPES::FIELD\_TYPE, 1228

VecAssemblyBegin  
  CMISS\_PETSC::interface, 807

VecAssemblyEnd  
  CMISS\_PETSC::interface, 807

VecCreate  
  CMISS\_PETSC::interface, 807

VecCreateGhost  
  CMISS\_PETSC::interface, 807

VecCreateGhostWithArray  
  CMISS\_PETSC::interface, 807

VecCreateMPI  
  CMISS\_PETSC::interface, 807

VecCreateMPIWithArray  
  CMISS\_PETSC::interface, 807

VecCreateSeq  
  CMISS\_PETSC::interface, 807

VecCreateSeqWithArray  
  CMISS\_PETSC::interface, 808

VecDestroy  
  CMISS\_PETSC::interface, 808

VecDuplicate  
  CMISS\_PETSC::interface, 808

VecGetArray

CMISS\_PETSC::interface, 808  
 VecGetArrayF90  
     CMISS\_PETSC::interface, 808  
 VecGetLocalSize  
     CMISS\_PETSC::interface, 808  
 VecGetOwnershipRange  
     CMISS\_PETSC::interface, 808  
 VecGetSize  
     CMISS\_PETSC::interface, 808  
 VecGetValues  
     CMISS\_PETSC::interface, 808  
 VecGhostGetLocalForm  
     CMISS\_PETSC::interface, 808  
 VecGhostRestoreLocalForm  
     CMISS\_PETSC::interface, 809  
 VecGhostUpdateBegin  
     CMISS\_PETSC::interface, 809  
 VecGhostUpdateEnd  
     CMISS\_PETSC::interface, 809  
 VecRestoreArray  
     CMISS\_PETSC::interface, 809  
 VecRestoreArrayF90  
     CMISS\_PETSC::interface, 809  
 VecSet  
     CMISS\_PETSC::interface, 809  
 VecSetFromOptions  
     CMISS\_PETSC::interface, 809  
 VecSetLocalToGlobalMapping  
     CMISS\_PETSC::interface, 809  
 VecSetSizes  
     CMISS\_PETSC::interface, 809  
 VecSetValue  
     CMISS\_PETSC::interface, 809  
 VecSetValuesLocal  
     CMISS\_PETSC::interface, 810  
**VECTOR**  
     TYPES::DISTRIBUTED\_VECTOR\_-  
         PETSC\_TYPE, 1080  
     TYPES::ELEMENT\_VECTOR\_TYPE, 1124  
     TYPES::EQUATIONS\_MATRICES\_RHS\_-  
         TYPE, 1155  
     TYPES::EQUATIONS\_MATRICES\_-  
         SOURCE\_TYPE, 1157  
**VECTOR\_ALL\_VALUES\_SET\_DP**  
     MATRIX\_VECTOR, 589  
     MATRIX\_VECTOR::VECTOR\_ALL\_-  
         VALUES\_SET, 995  
**VECTOR\_ALL\_VALUES\_SET\_INTG**  
     MATRIX\_VECTOR, 589  
     MATRIX\_VECTOR::VECTOR\_ALL\_-  
         VALUES\_SET, 995  
**VECTOR\_ALL\_VALUES\_SET\_L**  
     MATRIX\_VECTOR, 590  
     MATRIX\_VECTOR::VECTOR\_ALL\_-  
         VALUES\_SET, 995  
     VECTOR\_ALL\_VALUES\_SET\_SP  
         MATRIX\_VECTOR, 590  
         MATRIX\_VECTOR::VECTOR\_ALL\_-  
             VALUES\_SET, 996  
     VECTOR\_CREATE\_FINISH  
         MATRIX\_VECTOR, 591  
     VECTOR\_CREATE\_START  
         MATRIX\_VECTOR, 591  
     VECTOR\_DATA\_GET\_DP  
         MATRIX\_VECTOR, 591  
         MATRIX\_VECTOR::VECTOR\_DATA\_GET,  
             997  
     VECTOR\_DATA\_GET\_INTG  
         MATRIX\_VECTOR, 592  
         MATRIX\_VECTOR::VECTOR\_DATA\_GET,  
             997  
     VECTOR\_DATA\_GET\_L  
         MATRIX\_VECTOR, 592  
         MATRIX\_VECTOR::VECTOR\_DATA\_GET,  
             997  
     VECTOR\_DATA\_GET\_SP  
         MATRIX\_VECTOR, 593  
         MATRIX\_VECTOR::VECTOR\_DATA\_GET,  
             998  
     VECTOR\_DATA\_TYPE\_SET  
         MATRIX\_VECTOR, 593  
     VECTOR\_DESTROY  
         MATRIX\_VECTOR, 594  
     VECTOR\_DUPLICATE  
         MATRIX\_VECTOR, 594  
     VECTOR\_FINALISE  
         MATRIX\_VECTOR, 594  
     VECTOR\_FINISHED  
         TYPES::DISTRIBUTED\_VECTOR\_TYPE,  
             1086  
         TYPES::VECTOR\_TYPE, 1342  
     VECTOR\_INITIALISE  
         MATRIX\_VECTOR, 595  
     VECTOR\_SIZE\_SET  
         MATRIX\_VECTOR, 595  
     VECTOR\_VALUES\_GET\_DP  
         MATRIX\_VECTOR, 596  
         MATRIX\_VECTOR::VECTOR\_VALUES\_-  
             GET, 999  
     VECTOR\_VALUES\_GET\_DP1  
         MATRIX\_VECTOR, 596  
         MATRIX\_VECTOR::VECTOR\_VALUES\_-  
             GET, 999  
     VECTOR\_VALUES\_GET\_INTG  
         MATRIX\_VECTOR, 596  
         MATRIX\_VECTOR::VECTOR\_VALUES\_-  
             GET, 999

VECTOR\_VALUES\_GET\_INTG1  
  MATRIX\_VECTOR, 597  
  MATRIX\_VECTOR::VECTOR\_VALUES\_-  
    GET, 1000

VECTOR\_VALUES\_GET\_L  
  MATRIX\_VECTOR, 597  
  MATRIX\_VECTOR::VECTOR\_VALUES\_-  
    GET, 1000

VECTOR\_VALUES\_GET\_L1  
  MATRIX\_VECTOR, 598  
  MATRIX\_VECTOR::VECTOR\_VALUES\_-  
    GET, 1000

VECTOR\_VALUES\_GET\_SP  
  MATRIX\_VECTOR, 598  
  MATRIX\_VECTOR::VECTOR\_VALUES\_-  
    GET, 1001

VECTOR\_VALUES\_GET\_SP1  
  MATRIX\_VECTOR, 598  
  MATRIX\_VECTOR::VECTOR\_VALUES\_-  
    GET, 1001

VECTOR\_VALUES\_SET\_DP  
  MATRIX\_VECTOR, 599  
  MATRIX\_VECTOR::VECTOR\_VALUES\_-  
    SET, 1002

VECTOR\_VALUES\_SET\_DP1  
  MATRIX\_VECTOR, 599  
  MATRIX\_VECTOR::VECTOR\_VALUES\_-  
    SET, 1002

VECTOR\_VALUES\_SET\_INTG  
  MATRIX\_VECTOR, 600  
  MATRIX\_VECTOR::VECTOR\_VALUES\_-  
    SET, 1002

VECTOR\_VALUES\_SET\_INTG1  
  MATRIX\_VECTOR, 600  
  MATRIX\_VECTOR::VECTOR\_VALUES\_-  
    SET, 1003

VECTOR\_VALUES\_SET\_L  
  MATRIX\_VECTOR, 600  
  MATRIX\_VECTOR::VECTOR\_VALUES\_-  
    SET, 1003

VECTOR\_VALUES\_SET\_L1  
  MATRIX\_VECTOR, 601  
  MATRIX\_VECTOR::VECTOR\_VALUES\_-  
    SET, 1003

VECTOR\_VALUES\_SET\_SP  
  MATRIX\_VECTOR, 601  
  MATRIX\_VECTOR::VECTOR\_VALUES\_-  
    SET, 1004

VECTOR\_VALUES\_SET\_SP1  
  MATRIX\_VECTOR, 602  
  MATRIX\_VECTOR::VECTOR\_VALUES\_-  
    SET, 1004

VecView  
  CMISS\_PETSC::interface, 810

verify\_CH\_VS  
  ISO\_VARYING\_STRING, 511  
  ISO\_VARYING\_STRING::verify, 937

verify\_VS\_CH  
  ISO\_VARYING\_STRING, 511  
  ISO\_VARYING\_STRING::verify, 937

verify\_VS\_VS  
  ISO\_VARYING\_STRING, 511  
  ISO\_VARYING\_STRING::verify, 937

VOLUMES  
  TYPES::FIELD\_GEOMETRIC\_-  
    PARAMETERS\_TYPE, 1204

VSTRING\_TO\_LOWERCASE\_C  
  STRINGS, 744  
  STRINGS::VSTRING\_TO\_LOWERCASE,  
    1033

VSTRING\_TO\_LOWERCASE\_VS  
  STRINGS, 744  
  STRINGS::VSTRING\_TO\_LOWERCASE,  
    1033

VSTRING\_TO\_UPPERCASE\_C  
  STRINGS, 745  
  STRINGS::VSTRING\_TO\_UPPERCASE,  
    1034

VSTRING\_TO\_UPPERCASE\_VS  
  STRINGS, 745  
  STRINGS::VSTRING\_TO\_UPPERCASE,  
    1034

WR  
  INPUT\_OUTPUT, 432–452

WRITE\_BINARY\_FILE\_CHARACTER  
  BINARY\_FILE, 221  
  BINARY\_FILE::WRITE\_BINARY\_FILE,  
    791

WRITE\_BINARY\_FILE\_DP  
  BINARY\_FILE, 221  
  BINARY\_FILE::WRITE\_BINARY\_FILE,  
    791

WRITE\_BINARY\_FILE\_DP1  
  BINARY\_FILE, 221  
  BINARY\_FILE::WRITE\_BINARY\_FILE,  
    791

WRITE\_BINARY\_FILE\_DPC  
  BINARY\_FILE, 221  
  BINARY\_FILE::WRITE\_BINARY\_FILE,  
    791

WRITE\_BINARY\_FILE\_DPC1  
  BINARY\_FILE, 222  
  BINARY\_FILE::WRITE\_BINARY\_FILE,  
    792

WRITE\_BINARY\_FILE\_INTG  
  BINARY\_FILE, 222

BINARY\_FILE::WRITE\_BINARY\_FILE,  
    792

WRITE\_BINARY\_FILE\_INTG1  
    BINARY\_FILE, 222

    BINARY\_FILE::WRITE\_BINARY\_FILE,  
        792

WRITE\_BINARY\_FILE\_LINTG  
    BINARY\_FILE, 222

    BINARY\_FILE::WRITE\_BINARY\_FILE,  
        792

WRITE\_BINARY\_FILE\_LINTG1  
    BINARY\_FILE, 222

    BINARY\_FILE::WRITE\_BINARY\_FILE,  
        792

WRITE\_BINARY\_FILE\_LOGICAL  
    BINARY\_FILE, 223

    BINARY\_FILE::WRITE\_BINARY\_FILE,  
        792

WRITE\_BINARY\_FILE\_LOGICAL1  
    BINARY\_FILE, 223

    BINARY\_FILE::WRITE\_BINARY\_FILE,  
        792

WRITE\_BINARY\_FILE\_SINTG  
    BINARY\_FILE, 223

    BINARY\_FILE::WRITE\_BINARY\_FILE,  
        793

WRITE\_BINARY\_FILE\_SINTG1  
    BINARY\_FILE, 223

    BINARY\_FILE::WRITE\_BINARY\_FILE,  
        793

WRITE\_BINARY\_FILE\_SP  
    BINARY\_FILE, 224

    BINARY\_FILE::WRITE\_BINARY\_FILE,  
        793

WRITE\_BINARY\_FILE\_SPC1  
    BINARY\_FILE, 224

    BINARY\_FILE::WRITE\_BINARY\_FILE,  
        793

WRITE\_BINARY\_TAG\_HEADER  
    BINARY\_FILE, 224

WRITE\_STR  
    BASE\_ROUTINES, 207

WRITE\_STRING\_C  
    INPUT\_OUTPUT, 453

    INPUT\_OUTPUT::WRITE\_STRING, 868

WRITE\_STRING\_FMT  
    INPUT\_OUTPUT, 453–457

    INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
        TWO\_VALUE, 871

WRITE\_STRING\_FMT\_TWO\_VALUE\_C\_C  
    INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
        TWO\_VALUE, 871

WRITE\_STRING\_FMT\_TWO\_VALUE\_C\_DP  
    INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
        TWO\_VALUE, 871

WRITE\_STRING\_FMT\_TWO\_VALUE\_C\_INTG  
    INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
        TWO\_VALUE, 871

WRITE\_STRING\_FMT\_TWO\_VALUE\_C\_L  
    INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
        TWO\_VALUE, 871

WRITE\_STRING\_FMT\_TWO\_VALUE\_C\_SP  
    INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
        TWO\_VALUE, 871

WRITE\_STRING\_FMT\_TWO\_VALUE\_C\_VS  
    INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
        TWO\_VALUE, 871

WRITE\_STRING\_FMT\_TWO\_VALUE\_DP\_C  
    INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
        TWO\_VALUE, 871

WRITE\_STRING\_FMT\_TWO\_VALUE\_DP\_DP  
    INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
        TWO\_VALUE, 871

WRITE\_STRING\_FMT\_TWO\_VALUE\_DP\_-  
    INTG  
        INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
            TWO\_VALUE, 871

WRITE\_STRING\_FMT\_TWO\_VALUE\_DP\_L  
    INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
        TWO\_VALUE, 871

WRITE\_STRING\_FMT\_TWO\_VALUE\_DP\_SP  
    INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
        TWO\_VALUE, 871

WRITE\_STRING\_FMT\_TWO\_VALUE\_DP\_VS  
    INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
        TWO\_VALUE, 871

WRITE\_STRING\_FMT\_TWO\_VALUE\_INTG\_C  
    INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
        TWO\_VALUE, 871

WRITE\_STRING\_FMT\_TWO\_VALUE\_INTG\_-  
    DP  
        INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
            TWO\_VALUE, 871

WRITE\_STRING\_FMT\_TWO\_VALUE\_INTG\_-  
    INTG  
        INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
            TWO\_VALUE, 871

WRITE\_STRING\_FMT\_TWO\_VALUE\_INTG\_L  
    INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
        TWO\_VALUE, 871

WRITE\_STRING\_FMT\_TWO\_VALUE\_INTG\_-  
    SP

INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
TWO\_VALUE, 871  
WRITE\_STRING\_FMT\_TWO\_VALUE\_INTG\_-  
VS  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
TWO\_VALUE, 871  
WRITE\_STRING\_FMT\_TWO\_VALUE\_L\_C  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
TWO\_VALUE, 871  
WRITE\_STRING\_FMT\_TWO\_VALUE\_L\_DP  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
TWO\_VALUE, 871  
WRITE\_STRING\_FMT\_TWO\_VALUE\_L\_INTG  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
TWO\_VALUE, 871  
WRITE\_STRING\_FMT\_TWO\_VALUE\_L\_L  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
TWO\_VALUE, 871  
WRITE\_STRING\_FMT\_TWO\_VALUE\_L\_SP  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
TWO\_VALUE, 871  
WRITE\_STRING\_FMT\_TWO\_VALUE\_L\_VS  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
TWO\_VALUE, 871  
WRITE\_STRING\_FMT\_TWO\_VALUE\_SP\_C  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
TWO\_VALUE, 871  
WRITE\_STRING\_FMT\_TWO\_VALUE\_SP\_DP  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
TWO\_VALUE, 871  
WRITE\_STRING\_FMT\_TWO\_VALUE\_SP\_-  
INTG  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
TWO\_VALUE, 871  
WRITE\_STRING\_FMT\_TWO\_VALUE\_SP\_L  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
TWO\_VALUE, 871  
WRITE\_STRING\_FMT\_TWO\_VALUE\_SP\_SP  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
TWO\_VALUE, 871  
WRITE\_STRING\_FMT\_TWO\_VALUE\_SP\_VS  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
TWO\_VALUE, 871  
WRITE\_STRING\_FMT\_TWO\_VALUE\_VS\_C  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
TWO\_VALUE, 871  
WRITE\_STRING\_FMT\_TWO\_VALUE\_VS\_DP  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
TWO\_VALUE, 871  
WRITE\_STRING\_FMT\_TWO\_VALUE\_VS\_-  
INTG  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
TWO\_VALUE, 871  
WRITE\_STRING\_FMT\_TWO\_VALUE\_VS\_L

INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
TWO\_VALUE, 871  
WRITE\_STRING\_FMT\_TWO\_VALUE\_VS\_SP  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
TWO\_VALUE, 871  
WRITE\_STRING\_FMT\_TWO\_VALUE\_VS\_VS  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
TWO\_VALUE, 871  
WRITE\_STRING\_FMT\_VALUE\_C  
INPUT\_OUTPUT, 458  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
VALUE, 872  
WRITE\_STRING\_FMT\_VALUE\_DP  
INPUT\_OUTPUT, 458  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
VALUE, 872  
WRITE\_STRING\_FMT\_VALUE\_INTG  
INPUT\_OUTPUT, 459  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
VALUE, 873  
WRITE\_STRING\_FMT\_VALUE\_L  
INPUT\_OUTPUT, 459  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
VALUE, 873  
WRITE\_STRING\_FMT\_VALUE\_LINTG  
INPUT\_OUTPUT, 460  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
VALUE, 874  
WRITE\_STRING\_FMT\_VALUE\_SP  
INPUT\_OUTPUT, 460  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
VALUE, 874  
WRITE\_STRING\_FMT\_VALUE\_VS  
INPUT\_OUTPUT, 461  
INPUT\_OUTPUT::WRITE\_STRING\_FMT\_-  
VALUE, 874  
WRITE\_STRING\_IDX  
INPUT\_OUTPUT, 461–465  
WRITE\_STRING\_IDX\_VECTOR\_DP  
INPUT\_OUTPUT::WRITE\_STRING\_IDX\_-  
VECTOR, 876  
WRITE\_STRING\_IDX\_VECTOR\_INTG  
INPUT\_OUTPUT::WRITE\_STRING\_IDX\_-  
VECTOR, 876  
WRITE\_STRING\_IDX\_VECTOR\_L  
INPUT\_OUTPUT::WRITE\_STRING\_IDX\_-  
VECTOR, 876  
WRITE\_STRING\_IDX\_VECTOR\_LINTG  
INPUT\_OUTPUT::WRITE\_STRING\_IDX\_-  
VECTOR, 876  
WRITE\_STRING\_IDX\_VECTOR\_SP  
INPUT\_OUTPUT::WRITE\_STRING\_IDX\_-  
VECTOR, 876  
WRITE\_STRING\_MATRIX\_DP

INPUT\_OUTPUT, 465  
 INPUT\_OUTPUT::WRITE\_STRING\_-  
     MATRIX, 877  
 WRITE\_STRING\_MATRIX\_INTG  
     INPUT\_OUTPUT, 467  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         MATRIX, 878  
 WRITE\_STRING\_MATRIX\_L  
     INPUT\_OUTPUT, 468  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         MATRIX, 879  
 WRITE\_STRING\_MATRIX\_LINTG  
     INPUT\_OUTPUT, 469  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         MATRIX, 880  
 WRITE\_STRING\_MATRIX\_NAME\_AND\_-  
     INDICES  
     INPUT\_OUTPUT\_MatrixNameIndexFormat,  
         90  
 WRITE\_STRING\_MATRIX\_NAME\_ONLY  
     INPUT\_OUTPUT\_MatrixNameIndexFormat,  
         90  
 WRITE\_STRING\_MATRIX\_SP  
     INPUT\_OUTPUT, 470  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         MATRIX, 880  
 WRITE\_STRING\_TWO\_VALUE\_C\_C  
     INPUT\_OUTPUT, 471  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         TWO\_VALUE, 883  
 WRITE\_STRING\_TWO\_VALUE\_C\_DP  
     INPUT\_OUTPUT, 472  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         TWO\_VALUE, 884  
 WRITE\_STRING\_TWO\_VALUE\_C\_INTG  
     INPUT\_OUTPUT, 473  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         TWO\_VALUE, 884  
 WRITE\_STRING\_TWO\_VALUE\_C\_L  
     INPUT\_OUTPUT, 473  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         TWO\_VALUE, 884  
 WRITE\_STRING\_TWO\_VALUE\_C\_SP  
     INPUT\_OUTPUT, 474  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         TWO\_VALUE, 885  
 WRITE\_STRING\_TWO\_VALUE\_C\_VS  
     INPUT\_OUTPUT, 474  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         TWO\_VALUE, 885  
 WRITE\_STRING\_TWO\_VALUE\_DP\_C  
     INPUT\_OUTPUT, 475  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         TWO\_VALUE, 886  
 WRITE\_STRING\_TWO\_VALUE\_DP\_DP  
     INPUT\_OUTPUT, 476  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         TWO\_VALUE, 886  
 WRITE\_STRING\_TWO\_VALUE\_DP\_INTG  
     INPUT\_OUTPUT, 476  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         TWO\_VALUE, 887  
 WRITE\_STRING\_TWO\_VALUE\_DP\_L  
     INPUT\_OUTPUT, 477  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         TWO\_VALUE, 887  
 WRITE\_STRING\_TWO\_VALUE\_DP\_SP  
     INPUT\_OUTPUT, 477  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         TWO\_VALUE, 888  
 WRITE\_STRING\_TWO\_VALUE\_DP\_VS  
     INPUT\_OUTPUT, 478  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         TWO\_VALUE, 888  
 WRITE\_STRING\_TWO\_VALUE\_INTG\_C  
     INPUT\_OUTPUT, 478  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         TWO\_VALUE, 889  
 WRITE\_STRING\_TWO\_VALUE\_INTG\_DP  
     INPUT\_OUTPUT, 479  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         TWO\_VALUE, 889  
 WRITE\_STRING\_TWO\_VALUE\_INTG\_INTG  
     INPUT\_OUTPUT, 480  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         TWO\_VALUE, 890  
 WRITE\_STRING\_TWO\_VALUE\_INTG\_L  
     INPUT\_OUTPUT, 480  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         TWO\_VALUE, 890  
 WRITE\_STRING\_TWO\_VALUE\_INTG\_SP  
     INPUT\_OUTPUT, 481  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         TWO\_VALUE, 891  
 WRITE\_STRING\_TWO\_VALUE\_INTG\_VS  
     INPUT\_OUTPUT, 481  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         TWO\_VALUE, 891  
 WRITE\_STRING\_TWO\_VALUE\_L\_C  
     INPUT\_OUTPUT, 482  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         TWO\_VALUE, 892  
 WRITE\_STRING\_TWO\_VALUE\_L\_DP  
     INPUT\_OUTPUT, 483  
     INPUT\_OUTPUT::WRITE\_STRING\_-  
         TWO\_VALUE, 892  
 WRITE\_STRING\_TWO\_VALUE\_L\_INTG  
     INPUT\_OUTPUT, 483

INPUT\_OUTPUT::WRITE\_STRING\_-  
TWO\_VALUE, 893  
WRITE\_STRING\_TWO\_VALUE\_L\_L  
INPUT\_OUTPUT, 484  
INPUT\_OUTPUT::WRITE\_STRING\_-  
TWO\_VALUE, 893  
WRITE\_STRING\_TWO\_VALUE\_L\_SP  
INPUT\_OUTPUT, 484  
INPUT\_OUTPUT::WRITE\_STRING\_-  
TWO\_VALUE, 894  
WRITE\_STRING\_TWO\_VALUE\_L\_VS  
INPUT\_OUTPUT, 485  
INPUT\_OUTPUT::WRITE\_STRING\_-  
TWO\_VALUE, 894  
WRITE\_STRING\_TWO\_VALUE\_SP\_C  
INPUT\_OUTPUT, 485  
INPUT\_OUTPUT::WRITE\_STRING\_-  
TWO\_VALUE, 895  
WRITE\_STRING\_TWO\_VALUE\_SP\_DP  
INPUT\_OUTPUT, 486  
INPUT\_OUTPUT::WRITE\_STRING\_-  
TWO\_VALUE, 895  
WRITE\_STRING\_TWO\_VALUE\_SP\_INTG  
INPUT\_OUTPUT, 487  
INPUT\_OUTPUT::WRITE\_STRING\_-  
TWO\_VALUE, 896  
WRITE\_STRING\_TWO\_VALUE\_SP\_L  
INPUT\_OUTPUT, 487  
INPUT\_OUTPUT::WRITE\_STRING\_-  
TWO\_VALUE, 896  
WRITE\_STRING\_TWO\_VALUE\_SP\_SP  
INPUT\_OUTPUT, 488  
INPUT\_OUTPUT::WRITE\_STRING\_-  
TWO\_VALUE, 897  
WRITE\_STRING\_TWO\_VALUE\_SP\_VS  
INPUT\_OUTPUT, 488  
INPUT\_OUTPUT::WRITE\_STRING\_-  
TWO\_VALUE, 897  
WRITE\_STRING\_TWO\_VALUE\_VS\_C  
INPUT\_OUTPUT, 489  
INPUT\_OUTPUT::WRITE\_STRING\_-  
TWO\_VALUE, 898  
WRITE\_STRING\_TWO\_VALUE\_VS\_DP  
INPUT\_OUTPUT, 490  
INPUT\_OUTPUT::WRITE\_STRING\_-  
TWO\_VALUE, 898  
WRITE\_STRING\_TWO\_VALUE\_VS\_INTG  
INPUT\_OUTPUT, 490  
INPUT\_OUTPUT::WRITE\_STRING\_-  
TWO\_VALUE, 899  
WRITE\_STRING\_TWO\_VALUE\_VS\_L  
INPUT\_OUTPUT, 491  
INPUT\_OUTPUT::WRITE\_STRING\_-  
TWO\_VALUE, 899  
WRITE\_STRING\_TWO\_VALUE\_VS\_SP  
INPUT\_OUTPUT, 491  
INPUT\_OUTPUT::WRITE\_STRING\_-  
TWO\_VALUE, 900  
WRITE\_STRING\_TWO\_VALUE\_VS\_VS  
INPUT\_OUTPUT, 492  
INPUT\_OUTPUT::WRITE\_STRING\_-  
TWO\_VALUE, 900  
WRITE\_STRING\_VALUE\_C  
INPUT\_OUTPUT, 493  
INPUT\_OUTPUT::WRITE\_STRING\_-  
VALUE, 902  
WRITE\_STRING\_VALUE\_DP  
INPUT\_OUTPUT, 493  
INPUT\_OUTPUT::WRITE\_STRING\_-  
VALUE, 902  
WRITE\_STRING\_VALUE\_INTG  
INPUT\_OUTPUT, 494  
INPUT\_OUTPUT::WRITE\_STRING\_-  
VALUE, 903  
WRITE\_STRING\_VALUE\_L  
INPUT\_OUTPUT, 494  
INPUT\_OUTPUT::WRITE\_STRING\_-  
VALUE, 903  
WRITE\_STRING\_VALUE\_LINTG  
INPUT\_OUTPUT, 495  
INPUT\_OUTPUT::WRITE\_STRING\_-  
VALUE, 903  
WRITE\_STRING\_VALUE\_SP  
INPUT\_OUTPUT, 495  
INPUT\_OUTPUT::WRITE\_STRING\_-  
VALUE, 904  
WRITE\_STRING\_VALUE\_VS  
INPUT\_OUTPUT, 496  
INPUT\_OUTPUT::WRITE\_STRING\_-  
VALUE, 904  
WRITE\_STRING\_VECTOR\_DP  
INPUT\_OUTPUT::WRITE\_STRING\_-  
VECTOR, 905  
WRITE\_STRING\_VECTOR\_INTG  
INPUT\_OUTPUT::WRITE\_STRING\_-  
VECTOR, 905  
WRITE\_STRING\_VECTOR\_L  
INPUT\_OUTPUT::WRITE\_STRING\_-  
VECTOR, 905  
WRITE\_STRING\_VECTOR\_LINTG  
INPUT\_OUTPUT::WRITE\_STRING\_-  
VECTOR, 905  
WRITE\_STRING\_VECTOR\_SP  
INPUT\_OUTPUT::WRITE\_STRING\_-  
VECTOR, 905  
WRITE\_STRING\_VS  
INPUT\_OUTPUT, 496  
INPUT\_OUTPUT::WRITE\_STRING, 868

XI\_DIRECTION  
  TYPES::DECOMPOSITION\_LINE\_TYPE,  
    [1060](#)  
XI\_DIRECTION1  
  TYPES::DOMAIN\_FACE\_TYPE, [1097](#)  
XI\_DIRECTION2  
  TYPES::DOMAIN\_FACE\_TYPE, [1097](#)