

CWIPi

Generated by Doxygen 1.9.3



<b>1 Data Type Index</b>	<b>1</b>
1.1 Data Types List	1
<b>2 File Index</b>	<b>3</b>
2.1 File List	3
<b>3 Data Type Documentation</b>	<b>5</b>
3.1 cwp::cwp_c_to_f_string Interface Reference	5
3.1.1 Member Function/Subroutine Documentation	5
3.1.1.1 cwp_c_to_f_string_()	5
3.2 cwp::cwp_codes_list_get Interface Reference	6
3.2.1 Member Function/Subroutine Documentation	6
3.2.1.1 cwp_codes_list_get_()	6
3.3 cwp::CWP_Codes_nb_get Interface Reference	6
3.3.1 Detailed Description	6
3.4 cwp::cwp_computed_tgts_bcast_enable Interface Reference	7
3.4.1 Member Function/Subroutine Documentation	7
3.4.1.1 cwp_computed_tgts_bcast_enable_()	7
3.5 cwp::cwp_computed_tgts_get Interface Reference	7
3.5.1 Member Function/Subroutine Documentation	7
3.5.1.1 cwp_computed_tgts_get_()	8
3.6 cwp::cwp_cpl_barrier Interface Reference	8
3.6.1 Member Function/Subroutine Documentation	8
3.6.1.1 cwp_cpl_barrier_()	8
3.7 cwp::cwp_cpl_create Interface Reference	9
3.7.1 Member Function/Subroutine Documentation	9
3.7.1.1 cwp_cpl_create_()	9
3.8 cwp::cwp_cpl_del Interface Reference	10
3.8.1 Member Function/Subroutine Documentation	10
3.8.1.1 cwp_cpl_del_()	10
3.9 cwp::cwp_cpl_spatial_interp_algo_get Interface Reference	10
3.9.1 Member Function/Subroutine Documentation	10
3.9.1.1 cwp_cpl_spatial_interp_algo_get_()	11
3.10 cwp::cwp_field_create Interface Reference	11
3.10.1 Member Function/Subroutine Documentation	11
3.10.1.1 cwp_field_create_()	11
3.11 cwp::cwp_field_data_set Interface Reference	12
3.11.1 Member Function/Subroutine Documentation	12
3.11.1.1 cwp_field_data_set_()	12
3.12 cwp::cwp_field_del Interface Reference	13
3.12.1 Member Function/Subroutine Documentation	13
3.12.1.1 cwp_field_del_()	13
3.13 cwp::cwp_field_dof_location_get Interface Reference	13

3.13.1 Member Function/Subroutine Documentation	13
3.13.1.1 cwp_field_dof_location_get_()	14
3.14 cwp::cwp_field_interp_function_set Interface Reference	14
3.14.1 Member Function/Subroutine Documentation	14
3.14.1.1 cwp_field_interp_function_set_()	14
3.15 cwp::cwp_field_interp_function_unset Interface Reference	15
3.15.1 Member Function/Subroutine Documentation	15
3.15.1.1 cwp_field_interp_function_unset_()	15
3.16 cwp::cwp_field_intersection_tgt_elt_volumes_get Interface Reference	16
3.16.1 Member Function/Subroutine Documentation	16
3.16.1.1 cwp_field_intersection_tgt_elt_volumes_get_()	16
3.17 cwp::cwp_field_intersection_volumes_get Interface Reference	16
3.17.1 Member Function/Subroutine Documentation	16
3.17.1.1 cwp_field_intersection_volumes_get_()	17
3.18 cwp::cwp_field_irecv Interface Reference	17
3.18.1 Member Function/Subroutine Documentation	17
3.18.1.1 cwp_field_irecv_()	17
3.19 cwp::cwp_field_issend Interface Reference	18
3.19.1 Member Function/Subroutine Documentation	18
3.19.1.1 cwp_field_issend_()	18
3.20 cwp::cwp_field_location_internal_cell_vtx_get Interface Reference	19
3.20.1 Member Function/Subroutine Documentation	19
3.20.1.1 cwp_field_location_internal_cell_vtx_get_()	19
3.21 cwp::cwp_field_location_point_data_get Interface Reference	19
3.21.1 Member Function/Subroutine Documentation	20
3.21.1.1 cwp_field_location_point_data_get_()	20
3.22 cwp::cwp_field_location_weights_get Interface Reference	20
3.22.1 Member Function/Subroutine Documentation	20
3.22.1.1 cwp_field_location_weights_get_()	21
3.23 cwp::cwp_field_n_components_get Interface Reference	21
3.23.1 Member Function/Subroutine Documentation	21
3.23.1.1 cwp_field_n_components_get_()	21
3.24 cwp::cwp_field_n_dof_get Interface Reference	22
3.24.1 Member Function/Subroutine Documentation	22
3.24.1.1 cwp_field_n_dof_get_()	22
3.25 cwp::cwp_field_nearest_neighbors_coord_get Interface Reference	22
3.25.1 Member Function/Subroutine Documentation	23
3.25.1.1 cwp_field_nearest_neighbors_coord_get_()	23
3.26 cwp::cwp_field_nearest_neighbors_distances_get Interface Reference	23
3.26.1 Member Function/Subroutine Documentation	23
3.26.1.1 cwp_field_nearest_neighbors_distances_get_()	23
3.27 cwp::cwp_field_src_data_properties_get Interface Reference	24

3.27.1 Member Function/Subroutine Documentation	24
3.27.1.1 cwp_field_src_data_properties_get_()	24
3.28 cwp::cwp_field_storage_get Interface Reference	25
3.28.1 Member Function/Subroutine Documentation	25
3.28.1.1 cwp_field_storage_get_()	25
3.29 cwp::cwp_field_tgt_data_properties_get Interface Reference	25
3.29.1 Member Function/Subroutine Documentation	26
3.29.1.1 cwp_field_tgt_data_properties_get_()	26
3.30 cwp::cwp_field_wait_irecv Interface Reference	26
3.30.1 Member Function/Subroutine Documentation	26
3.30.1.1 cwp_field_wait_irecv_()	27
3.31 cwp::cwp_field_wait_issend Interface Reference	27
3.31.1 Member Function/Subroutine Documentation	27
3.31.1.1 cwp_field_wait_issend_()	27
3.32 cwp::CWP_Finalize Interface Reference	28
3.32.1 Detailed Description	28
3.33 cwp::cwp_global_data_irecv Interface Reference	28
3.33.1 Member Function/Subroutine Documentation	28
3.33.1.1 cwp_global_data_irecv_int()	28
3.34 cwp::cwp_global_data_issend Interface Reference	29
3.34.1 Member Function/Subroutine Documentation	29
3.34.1.1 cwp_global_data_issend_int()	29
3.35 cwp::cwp_global_data_wait_irecv Interface Reference	30
3.35.1 Member Function/Subroutine Documentation	30
3.35.1.1 cwp_global_data_wait_irecv_()	30
3.36 cwp::cwp_global_data_wait_issend Interface Reference	30
3.36.1 Member Function/Subroutine Documentation	30
3.36.1.1 cwp_global_data_wait_issend_()	30
3.37 cwp::cwp_init Interface Reference	31
3.37.1 Member Function/Subroutine Documentation	31
3.37.1.1 cwp_init_()	31
3.38 cwp::cwp_involved_srcs_bcast_enable Interface Reference	32
3.38.1 Member Function/Subroutine Documentation	32
3.38.1.1 cwp_involved_srcs_bcast_enable_()	32
3.39 cwp::cwp_involved_srcs_get Interface Reference	32
3.39.1 Member Function/Subroutine Documentation	32
3.39.1.1 cwp_involved_srcs_get_()	33
3.40 cwp::cwp_loc_codes_list_get Interface Reference	33
3.40.1 Member Function/Subroutine Documentation	33
3.40.1.1 cwp_loc_codes_list_get_()	33
3.41 cwp::CWP_Loc_codes_nb_get Interface Reference	34
3.41.1 Detailed Description	34

3.42 cwp::cwp_mesh_interf_block_add Interface Reference . . . . .	34
3.42.1 Member Function/Subroutine Documentation . . . . .	34
3.42.1.1 cwp_mesh_interf_block_add_() . . . . .	34
3.43 cwp::cwp_mesh_interf_block_std_get Interface Reference . . . . .	35
3.43.1 Member Function/Subroutine Documentation . . . . .	35
3.43.1.1 cwp_mesh_interf_block_std_get_() . . . . .	35
3.44 cwp::cwp_mesh_interf_block_std_set Interface Reference . . . . .	36
3.44.1 Member Function/Subroutine Documentation . . . . .	36
3.44.1.1 cwp_mesh_interf_block_std_set_() . . . . .	36
3.45 cwp::cwp_mesh_interf_c_poly_block_get Interface Reference . . . . .	37
3.45.1 Member Function/Subroutine Documentation . . . . .	37
3.45.1.1 cwp_mesh_interf_c_poly_block_get_() . . . . .	37
3.46 cwp::cwp_mesh_interf_c_poly_block_set Interface Reference . . . . .	38
3.46.1 Member Function/Subroutine Documentation . . . . .	38
3.46.1.1 cwp_mesh_interf_c_poly_block_set_() . . . . .	39
3.47 cwp::cwp_mesh_interf_del Interface Reference . . . . .	39
3.47.1 Member Function/Subroutine Documentation . . . . .	39
3.47.1.1 cwp_mesh_interf_del_() . . . . .	40
3.48 cwp::cwp_mesh_interf_f_poly_block_get Interface Reference . . . . .	40
3.48.1 Member Function/Subroutine Documentation . . . . .	40
3.48.1.1 cwp_mesh_interf_f_poly_block_get_() . . . . .	40
3.49 cwp::cwp_mesh_interf_f_poly_block_set Interface Reference . . . . .	41
3.49.1 Member Function/Subroutine Documentation . . . . .	41
3.49.1.1 cwp_mesh_interf_f_poly_block_set_() . . . . .	41
3.50 cwp::cwp_mesh_interf_finalize Interface Reference . . . . .	42
3.50.1 Member Function/Subroutine Documentation . . . . .	42
3.50.1.1 cwp_mesh_interf_finalize_() . . . . .	42
3.51 cwp::cwp_mesh_interf_from_cellface_set Interface Reference . . . . .	42
3.51.1 Member Function/Subroutine Documentation . . . . .	42
3.51.1.1 cwp_mesh_interf_from_cellface_set_() . . . . .	43
3.52 cwp::cwp_mesh_interf_from_faceedge_set Interface Reference . . . . .	43
3.52.1 Member Function/Subroutine Documentation . . . . .	43
3.52.1.1 cwp_mesh_interf_from_faceedge_set_() . . . . .	44
3.53 cwp::cwp_mesh_interf_vtx_set Interface Reference . . . . .	44
3.53.1 Member Function/Subroutine Documentation . . . . .	44
3.53.1.1 cwp_mesh_interf_vtx_set_() . . . . .	44
3.54 cwp::cwp_n_computed_tgts_get Interface Reference . . . . .	45
3.54.1 Member Function/Subroutine Documentation . . . . .	45
3.54.1.1 cwp_n_computed_tgts_get_() . . . . .	45
3.55 cwp::cwp_n_involved_srcs_get Interface Reference . . . . .	46
3.55.1 Member Function/Subroutine Documentation . . . . .	46
3.55.1.1 cwp_n_involved_srcs_get_() . . . . .	46

3.56 cwp::cwp_n_uncomputed_tgts_get Interface Reference . . . . .	46
3.56.1 Member Function/Subroutine Documentation . . . . .	47
3.56.1.1 cwp_n_uncomputed_tgts_get_() . . . . .	47
3.57 cwp::cwp_output_file_set Interface Reference . . . . .	47
3.57.1 Member Function/Subroutine Documentation . . . . .	47
3.57.1.1 cwp_output_file_set_() . . . . .	47
3.58 cwp::cwp_param_add Interface Reference . . . . .	48
3.58.1 Member Function/Subroutine Documentation . . . . .	48
3.58.1.1 cwp_param_add_int_() . . . . .	48
3.59 cwp::cwp_param_del Interface Reference . . . . .	48
3.59.1 Member Function/Subroutine Documentation . . . . .	49
3.59.1.1 cwp_param_del_() . . . . .	49
3.60 cwp::cwp_param_get Interface Reference . . . . .	49
3.60.1 Member Function/Subroutine Documentation . . . . .	49
3.60.1.1 cwp_param_get_int_() . . . . .	49
3.61 cwp::cwp_param_is Interface Reference . . . . .	50
3.61.1 Member Function/Subroutine Documentation . . . . .	50
3.61.1.1 cwp_param_is_() . . . . .	50
3.62 cwp::cwp_param_list_get Interface Reference . . . . .	51
3.62.1 Member Function/Subroutine Documentation . . . . .	51
3.62.1.1 cwp_param_list_get_() . . . . .	51
3.63 cwp::cwp_param_lock Interface Reference . . . . .	51
3.63.1 Member Function/Subroutine Documentation . . . . .	51
3.63.1.1 cwp_param_lock_() . . . . .	51
3.64 cwp::cwp_param_n_get Interface Reference . . . . .	52
3.64.1 Member Function/Subroutine Documentation . . . . .	52
3.64.1.1 cwp_param_n_get_() . . . . .	52
3.65 cwp::cwp_param_reduce Interface Reference . . . . .	52
3.65.1 Member Function/Subroutine Documentation . . . . .	53
3.65.1.1 cwp_param_reduce_int_() . . . . .	53
3.66 cwp::cwp_param_set Interface Reference . . . . .	53
3.66.1 Member Function/Subroutine Documentation . . . . .	53
3.66.1.1 cwp_param_set_int_() . . . . .	53
3.67 cwp::cwp_param_unlock Interface Reference . . . . .	54
3.67.1 Member Function/Subroutine Documentation . . . . .	54
3.67.1.1 cwp_param_unlock_() . . . . .	54
3.68 cwp::cwp_part_data_create Interface Reference . . . . .	54
3.68.1 Member Function/Subroutine Documentation . . . . .	55
3.68.1.1 cwp_part_data_create_() . . . . .	55
3.69 cwp::cwp_part_data_del Interface Reference . . . . .	55
3.69.1 Member Function/Subroutine Documentation . . . . .	55
3.69.1.1 cwp_part_data_del_() . . . . .	55

3.70 cwp::cwp_part_data_irecv Interface Reference	56
3.70.1 Member Function/Subroutine Documentation	56
3.70.1.1 cwp_part_data_irecv_()	56
3.71 cwp::cwp_part_data_issend Interface Reference	57
3.71.1 Member Function/Subroutine Documentation	57
3.71.1.1 cwp_part_data_issend_()	57
3.72 cwp::cwp_part_data_wait_irecv Interface Reference	57
3.72.1 Member Function/Subroutine Documentation	57
3.72.1.1 cwp_part_data_wait_irecv_()	58
3.73 cwp::cwp_part_data_wait_issend Interface Reference	58
3.73.1 Member Function/Subroutine Documentation	58
3.73.1.1 cwp_part_data_wait_issend_()	58
3.74 cwp::CWP_Properties_dump Interface Reference	59
3.74.1 Detailed Description	59
3.75 cwp::cwp_spatial_interp_property_set Interface Reference	59
3.75.1 Member Function/Subroutine Documentation	59
3.75.1.1 cwp_spatial_interp_property_set_()	59
3.76 cwp::cwp_spatial_interp_weights_compute Interface Reference	60
3.76.1 Member Function/Subroutine Documentation	60
3.76.1.1 cwp_spatial_interp_weights_compute_()	60
3.77 cwp::cwp_state_get Interface Reference	60
3.77.1 Member Function/Subroutine Documentation	61
3.77.1.1 cwp_state_get_()	61
3.78 cwp::cwp_state_update Interface Reference	61
3.78.1 Member Function/Subroutine Documentation	61
3.78.1.1 cwp_state_update_()	61
3.79 cwp::cwp_time_step_beg Interface Reference	62
3.79.1 Member Function/Subroutine Documentation	62
3.79.1.1 cwp_time_step_beg_()	62
3.80 cwp::cwp_time_step_end Interface Reference	62
3.80.1 Member Function/Subroutine Documentation	62
3.80.1.1 cwp_time_step_end_()	63
3.81 cwp::cwp_uncomputed_tgt_get Interface Reference	63
3.81.1 Member Function/Subroutine Documentation	63
3.81.1.1 cwp_uncomputed_tgt_get_()	63
3.82 cwp::cwp_user_structure_get Interface Reference	64
3.82.1 Member Function/Subroutine Documentation	64
3.82.1.1 cwp_user_structure_get_()	64
3.83 cwp::cwp_user_structure_set Interface Reference	64
3.84 cwp::cwp_user_tgt_pts_set Interface Reference	65
3.84.1 Member Function/Subroutine Documentation	65
3.84.1.1 cwp_user_tgt_pts_set_()	65



3.85 cwp::cwp_visu_set Interface Reference . . . . .	65
3.85.1 Member Function/Subroutine Documentation . . . . .	66
3.85.1.1 cwp_visu_set_() . . . . .	66
<b>4 File Documentation . . . . .</b>	<b>67</b>
4.1 fortran/new/cwp_f.f90 File Reference . . . . .	67
4.1.1 Function/Subroutine Documentation . . . . .	73
4.1.1.1 cwp_c_to_f_string_() . . . . .	74
4.1.1.2 cwp_codes_list_get_() . . . . .	75
4.1.1.3 cwp_computed_tgts_bcast_enable_() . . . . .	75
4.1.1.4 cwp_computed_tgts_get_() . . . . .	75
4.1.1.5 cwp_cpl_barrier_() . . . . .	77
4.1.1.6 cwp_cpl_create_() . . . . .	77
4.1.1.7 cwp_cpl_del_() . . . . .	78
4.1.1.8 cwp_cpl_spatial_interp_algo_get_() . . . . .	78
4.1.1.9 cwp_field_create_() . . . . .	78
4.1.1.10 cwp_field_data_set_() . . . . .	79
4.1.1.11 cwp_field_del_() . . . . .	79
4.1.1.12 cwp_field_dof_location_get_() . . . . .	80
4.1.1.13 cwp_field_interp_function_set_() . . . . .	80
4.1.1.14 cwp_field_interp_function_unset_() . . . . .	81
4.1.1.15 cwp_field_intersection_tgt_elt_volumes_get_() . . . . .	81
4.1.1.16 cwp_field_intersection_volumes_get_() . . . . .	81
4.1.1.17 cwp_field_irecv_() . . . . .	82
4.1.1.18 cwp_field_issend_() . . . . .	82
4.1.1.19 cwp_field_location_internal_cell_vtx_get_() . . . . .	83
4.1.1.20 cwp_field_location_point_data_get_() . . . . .	83
4.1.1.21 cwp_field_location_weights_get_() . . . . .	84
4.1.1.22 cwp_field_n_components_get_() . . . . .	84
4.1.1.23 cwp_field_n_dof_get_() . . . . .	84
4.1.1.24 cwp_field_nearest_neighbors_coord_get_() . . . . .	85
4.1.1.25 cwp_field_nearest_neighbors_distances_get_() . . . . .	85
4.1.1.26 cwp_field_src_data_properties_get_() . . . . .	86
4.1.1.27 cwp_field_storage_get_() . . . . .	86
4.1.1.28 cwp_field_tgt_data_properties_get_() . . . . .	86
4.1.1.29 cwp_field_wait_irecv_() . . . . .	87
4.1.1.30 cwp_field_wait_issend_() . . . . .	87
4.1.1.31 cwp_global_data_irecv_int() . . . . .	88
4.1.1.32 cwp_global_data_issend_int() . . . . .	88
4.1.1.33 cwp_global_data_wait_irecv_() . . . . .	88
4.1.1.34 cwp_global_data_wait_issend_() . . . . .	89
4.1.1.35 cwp_init_() . . . . .	89

4.1.1.36 cwp_involved_srcs_bcast_enable_()	90
4.1.1.37 cwp_involved_srcs_get_()	90
4.1.1.38 cwp_loc_codes_list_get_()	90
4.1.1.39 cwp_mesh_interf_block_add_()	91
4.1.1.40 cwp_mesh_interf_block_std_get_()	91
4.1.1.41 cwp_mesh_interf_block_std_set_()	92
4.1.1.42 cwp_mesh_interf_c_poly_block_get_()	93
4.1.1.43 cwp_mesh_interf_c_poly_block_set_()	93
4.1.1.44 cwp_mesh_interf_del_()	94
4.1.1.45 cwp_mesh_interf_f_poly_block_get_()	94
4.1.1.46 cwp_mesh_interf_f_poly_block_set_()	95
4.1.1.47 cwp_mesh_interf_finalize_()	96
4.1.1.48 cwp_mesh_interf_from_cellface_set_()	96
4.1.1.49 cwp_mesh_interf_from_faceedge_set_()	97
4.1.1.50 cwp_mesh_interf_vtx_set_()	97
4.1.1.51 cwp_n_computed_tgts_get_()	98
4.1.1.52 cwp_n_involved_srcs_get_()	98
4.1.1.53 cwp_n_uncomputed_tgts_get_()	98
4.1.1.54 cwp_output_file_set_()	99
4.1.1.55 cwp_output_fortran_unit_set_()	99
4.1.1.56 cwp_param_add_int_()	100
4.1.1.57 cwp_param_del_()	100
4.1.1.58 cwp_param_get_int_()	100
4.1.1.59 cwp_param_is_()	101
4.1.1.60 cwp_param_list_get_()	101
4.1.1.61 cwp_param_lock_()	101
4.1.1.62 cwp_param_n_get_()	102
4.1.1.63 cwp_param_reduce_int_()	102
4.1.1.64 cwp_param_set_int_()	102
4.1.1.65 cwp_param_unlock_()	103
4.1.1.66 cwp_part_data_create_()	103
4.1.1.67 cwp_part_data_del_()	104
4.1.1.68 cwp_part_data_irecv_()	104
4.1.1.69 cwp_part_data_issend_()	104
4.1.1.70 cwp_part_data_wait_irecv_()	105
4.1.1.71 cwp_part_data_wait_issend_()	105
4.1.1.72 cwp_spatial_interp_property_set_()	106
4.1.1.73 cwp_spatial_interp_weights_compute_()	106
4.1.1.74 cwp_state_get_()	106
4.1.1.75 cwp_state_update_()	107
4.1.1.76 cwp_time_step_beg_()	107
4.1.1.77 cwp_time_step_end_()	107

4.1.1.78 cwp_uncomputed_tgts_get_()	108
4.1.1.79 cwp_user_structure_get_()	108
4.1.1.80 cwp_user_tgt_pts_set_()	108
4.1.1.81 cwp_visu_set_()	109

<b>Index</b>	<b>111</b>
--------------	------------



# Chapter 1

## Data Type Index

### 1.1 Data Types List

Here are the data types with brief descriptions:

<a href="#">cwp::cwp_c_to_f_string</a>	5
<a href="#">cwp::cwp_codes_list_get</a>	6
<a href="#">cwp::CWP_Codes_nb_get</a>	
Return the number of codes known by CWIPI	6
<a href="#">cwp::cwp_computed_tgts_bcast_enable</a>	7
<a href="#">cwp::cwp_computed_tgts_get</a>	7
<a href="#">cwp::cwp_cpl_barrier</a>	8
<a href="#">cwp::cwp_cpl_create</a>	9
<a href="#">cwp::cwp_cpl_del</a>	10
<a href="#">cwp::cwp_cpl_spatial_interp_algo_get</a>	10
<a href="#">cwp::cwp_field_create</a>	11
<a href="#">cwp::cwp_field_data_set</a>	12
<a href="#">cwp::cwp_field_del</a>	13
<a href="#">cwp::cwp_field_dof_location_get</a>	13
<a href="#">cwp::cwp_field_interp_function_set</a>	14
<a href="#">cwp::cwp_field_interp_function_unset</a>	15
<a href="#">cwp::cwp_field_intersection_tgt_elt_volumes_get</a>	16
<a href="#">cwp::cwp_field_intersection_volumes_get</a>	16
<a href="#">cwp::cwp_field_irecv</a>	17
<a href="#">cwp::cwp_field_issend</a>	18
<a href="#">cwp::cwp_field_location_internal_cell_vtx_get</a>	19
<a href="#">cwp::cwp_field_location_point_data_get</a>	19
<a href="#">cwp::cwp_field_location_weights_get</a>	20
<a href="#">cwp::cwp_field_n_components_get</a>	21
<a href="#">cwp::cwp_field_n_dof_get</a>	22
<a href="#">cwp::cwp_field_nearest_neighbors_coord_get</a>	22
<a href="#">cwp::cwp_field_nearest_neighbors_distances_get</a>	23
<a href="#">cwp::cwp_field_src_data_properties_get</a>	24
<a href="#">cwp::cwp_field_storage_get</a>	25
<a href="#">cwp::cwp_field_tgt_data_properties_get</a>	25
<a href="#">cwp::cwp_field_wait_irecv</a>	26
<a href="#">cwp::cwp_field_wait_issend</a>	27
<a href="#">cwp::CWP_Finalize</a>	
Finalize CWIPI	28
<a href="#">cwp::cwp_global_data_irecv</a>	28

cwp::cwp_global_data_issend	29
cwp::cwp_global_data_wait_irecv	30
cwp::cwp_global_data_wait_issend	30
cwp::cwp_init	31
cwp::cwp_involved_srcs_bcast_enable	32
cwp::cwp_involved_srcs_get	32
cwp::cwp_loc_codes_list_get	33
cwp::CWP_Loc_codes_nb_get	
Return the number of local codes known by CWIPI	34
cwp::cwp_mesh_interf_block_add	34
cwp::cwp_mesh_interf_block_std_get	35
cwp::cwp_mesh_interf_block_std_set	36
cwp::cwp_mesh_interf_c_poly_block_get	37
cwp::cwp_mesh_interf_c_poly_block_set	38
cwp::cwp_mesh_interf_del	39
cwp::cwp_mesh_interf_f_poly_block_get	40
cwp::cwp_mesh_interf_f_poly_block_set	41
cwp::cwp_mesh_interf_finalize	42
cwp::cwp_mesh_interf_from_cellface_set	42
cwp::cwp_mesh_interf_from_faceedge_set	43
cwp::cwp_mesh_interf_vtx_set	44
cwp::cwp_n_computed_tgts_get	45
cwp::cwp_n_involved_srcs_get	46
cwp::cwp_n_uncomputed_tgts_get	46
cwp::cwp_output_file_set	47
cwp::cwp_param_add	48
cwp::cwp_param_del	48
cwp::cwp_param_get	49
cwp::cwp_param_is	50
cwp::cwp_param_list_get	51
cwp::cwp_param_lock	51
cwp::cwp_param_n_get	52
cwp::cwp_param_reduce	52
cwp::cwp_param_set	53
cwp::cwp_param_unlock	54
cwp::cwp_part_data_create	54
cwp::cwp_part_data_del	55
cwp::cwp_part_data_irecv	56
cwp::cwp_part_data_issend	57
cwp::cwp_part_data_wait_irecv	57
cwp::cwp_part_data_wait_issend	58
cwp::CWP_Properties_dump	
Dump code properties	59
cwp::cwp_spatial_interp_property_set	59
cwp::cwp_spatial_interp_weights_compute	60
cwp::cwp_state_get	60
cwp::cwp_state_update	61
cwp::cwp_time_step_beg	62
cwp::cwp_time_step_end	62
cwp::cwp_uncomputed_tgts_get	63
cwp::cwp_user_structure_get	64
cwp::cwp_user_structure_set	64
cwp::cwp_user_tgt_pts_set	65
cwp::cwp_visu_set	65

## Chapter 2

# File Index

### 2.1 File List

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

fortran/new/ <a href="#">cwp_f.f90</a> . . . . .	67
--	----





## Chapter 3

# Data Type Documentation

### 3.1 cwp::cwp\_c\_to\_f\_string Interface Reference

#### Public Member Functions

- `character(len=:)` function, pointer [cwp\\_c\\_to\\_f\\_string\\_](#) (`c_str`)  
*Create a Fortran string from a C string.*

#### 3.1.1 Member Function/Subroutine Documentation

##### 3.1.1.1 cwp\_c\_to\_f\_string\_()

```
character(len=:)  function, pointer cwp::cwp_c_to_f_string::cwp_c_to_f_string_ (  
    character(kind=c_char,len=1), dimension(*), intent(in) c_str )
```

Create a Fortran string from a C string.

This function creates a Fortran string from a C string. There is a string copy

#### Parameters

in	<i>c_str</i>	C string
----	--------------	----------

#### Returns

Fortran string

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

- [fortran/new/cwp\\_f.f90](#)

## 3.2 cwp::cwp\_codes\_list\_get Interface Reference

### Public Member Functions

- `character(256) function, dimension(:), allocatable cwp_codes_list_get_()`  
*Return list of codes known by CWIPI.*

### 3.2.1 Member Function/Subroutine Documentation

#### 3.2.1.1 cwp\_codes\_list\_get\_()

`character(256) function, dimension(:), allocatable cwp::cwp_codes_list_get::cwp_codes_list_get_`

Return list of codes known by CWIPI.

#### Returns

List of code names

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

- `fortran/new/cwp_f.f90`

## 3.3 cwp::CWP\_Codes\_nb\_get Interface Reference

Return the number of codes known by CWIPI.

### Public Member Functions

- `integer(c_int) function cwp_codes_nb_get()`

### 3.3.1 Detailed Description

Return the number of codes known by CWIPI.

#### Returns

Number of codes

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

- `fortran/new/cwp_f.f90`

## 3.4 cwp::cwp\_computed\_tgts\_bcast\_enable Interface Reference

### Public Member Functions

- subroutine [cwp\\_computed\\_tgts\\_bcast\\_enable](#) (local\_code\_name, cpl\_id, field\_id)  
*Enable broadcast of the computed targets ids (in CWP\_COMM\_PAR\_WITHOUT\_PART mode).*

### 3.4.1 Member Function/Subroutine Documentation

#### 3.4.1.1 cwp\_computed\_tgts\_bcast\_enable\_()

```
subroutine cwp::cwp_computed_tgts_bcast_enable::cwp_computed_tgts_bcast_enable_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id )
```

Enable broadcast of the computed targets ids (in CWP\_COMM\_PAR\_WITHOUT\_PART mode).

This function must be called in order for the computed targets to be accessible on non-root ranks

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Current partition

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

- fortran/new/[cwp\\_f.f90](#)

## 3.5 cwp::cwp\_computed\_tgts\_get Interface Reference

### Public Member Functions

- integer(c\_int) function, dimension(:), pointer [cwp\\_computed\\_tgts\\_get](#) (local\_code\_name, cpl\_id, field\_id, i←\_part)  
*Return computed targets.*

### 3.5.1 Member Function/Subroutine Documentation

### 3.5.1.1 cwp\_computed\_tgts\_get\_()

```
integer(c_int) function, dimension(:), pointer cwp::cwp_computed_tgts_get::cwp_computed_tgts←
_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part )
```

Return computed targets.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Current partition

#### Returns

Computed targets

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

- [fortran/new/cwp\\_f.f90](#)

## 3.6 cwp::cwp\_cpl\_barrier Interface Reference

### Public Member Functions

- subroutine [cwp\\_cpl\\_barrier\\_](#) (local\_code\_name, cpl\_id)  
*MPI Barrier on the coupling communicator.*

### 3.6.1 Member Function/Subroutine Documentation

#### 3.6.1.1 cwp\_cpl\_barrier\_()

```
subroutine cwp::cwp_cpl_barrier::cwp_cpl_barrier_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id )
```

MPI Barrier on the coupling communicator.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier

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

- [fortran/new/cwp\\_f.f90](#)

## 3.7 cwp::cwp\_cpl\_create Interface Reference

### Public Member Functions

- subroutine [cwp\\_cpl\\_create\\_](#) (*local\_code\_name*, *cpl\_id*, *coupled\_code\_name*, *entities\_dim*, *comm\_type*, *spatial\_interp*, *n\_part*, *displacement*, *freq*)  
*Create a coupling object and define its properties.*

### 3.7.1 Member Function/Subroutine Documentation

#### 3.7.1.1 cwp\_cpl\_create\_()

```

subroutine cwp::cwp_cpl_create::cwp_cpl_create_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) coupled_code_name,
    integer(kind = c_int) entities_dim,
    integer(kind = c_int) comm_type,
    integer(kind = c_int) spatial_interp,
    integer(kind = c_int) n_part,
    integer(kind = c_int) displacement,
    integer(kind = c_int) freq )

```

Create a coupling object and define its properties.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>coupled_code_name</i>	Distant or local coupled code name
in	<i>comm_type</i>	Communication type
in	<i>spatial_interp</i>	Spatial interpolation method
in	<i>n_part</i>	Number of interface partition
in	<i>displacement</i>	Mesh moving status
in	<i>recv_freq_type</i>	Type of receiving frequency

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

- [fortran/new/cwp\\_f.f90](#)

## 3.8 cwp::cwp\_cpl\_del Interface Reference

### Public Member Functions

- subroutine [cwp\\_cpl\\_del\\_](#) (local\_code\_name, cpl\_id)  
*Delete a coupling object.*

### 3.8.1 Member Function/Subroutine Documentation

#### 3.8.1.1 cwp\_cpl\_del\_()

```
subroutine cwp::cwp_cpl_del::cwp_cpl_del_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id )
```

Delete a coupling object.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier

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

- [fortran/new/cwp\\_f.f90](#)

## 3.9 cwp::cwp\_cpl\_spatial\_interp\_algo\_get Interface Reference

### Public Member Functions

- integer(c\_int) function [cwp\\_cpl\\_spatial\\_interp\\_algo\\_get\\_](#) (local\_code\_name, cpl\_id)  
*Get the coupling spatial interpolation algorithm.*

### 3.9.1 Member Function/Subroutine Documentation

### 3.9.1.1 cwp\_cpl\_spatial\_interp\_algo\_get\_()

```
integer(c_int) function cwp::cwp_cpl_spatial_interp_algo_get::cwp_cpl_spatial_interp_algo_↵
get_ (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id )
```

Get the coupling spatial interpolation algorithm.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier

#### Returns

Spatial interpolation method

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

- [fortran/new/cwp\\_f.f90](#)

## 3.10 cwp::cwp\_field\_create Interface Reference

### Public Member Functions

- subroutine [cwp\\_field\\_create\\_](#) (local\_code\_name, cpl\_id, field\_id, data\_type, storage, n\_component, target\_↵  
\_location, exch\_type, visu\_status)  
*Create a new field.*

### 3.10.1 Member Function/Subroutine Documentation

#### 3.10.1.1 cwp\_field\_create\_()

```
subroutine cwp::cwp_field_create::cwp_field_create_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) data_type,
    integer(c_int) storage,
    integer(c_int) n_component,
    integer(c_int) target_location,
    integer(c_int) exch_type,
    integer(c_int) visu_status )
```

Create a new field.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field id
in	<i>data_type</i>	Data type
in	<i>storage</i>	Storage type
in	<i>n_component</i>	Number of component
in	<i>target_location</i>	Target location
in	<i>exch_type</i>	Exchange type
in	<i>visu_status</i>	Visualization status

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

- [fortran/new/cwp\\_f.f90](#)

## 3.11 cwp::cwp\_field\_data\_set Interface Reference

### Public Member Functions

- subroutine [cwp\\_field\\_data\\_set\\_](#) (*local\_code\_name*, *cpl\_id*, *field\_id*, *i\_part*, *map\_type*, *data*)  
*Set field data.*

### 3.11.1 Member Function/Subroutine Documentation

#### 3.11.1.1 cwp\_field\_data\_set\_()

```
subroutine cwp::cwp_field_data_set::cwp_field_data_set_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part,
    integer(c_int) map_type,
    double precision, dimension(:), pointer data )
```

Set field data.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Current partition
in	<i>data_type</i>	Choice if data is set for the source or the target
in	<i>data</i>	Storage array (Mapping)



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

- [fortran/new/cwp\\_f.f90](#)

## 3.12 cwp::cwp\_field\_del Interface Reference

### Public Member Functions

- subroutine [cwp\\_field\\_del\\_](#) (local\_code\_name, cpl\_id, field\_id)  
*Delete a field.*

### 3.12.1 Member Function/Subroutine Documentation

#### 3.12.1.1 cwp\_field\_del\_()

```
subroutine cwp::cwp_field_del::cwp_field_del_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id )
```

Delete a field.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier

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

- [fortran/new/cwp\\_f.f90](#)

## 3.13 cwp::cwp\_field\_dof\_location\_get Interface Reference

### Public Member Functions

- integer(c\_int) function [cwp\\_field\\_dof\\_location\\_get\\_](#) (local\_code\_name, cpl\_id, field\_id)  
*Get target degrees of freedom location.*

### 3.13.1 Member Function/Subroutine Documentation

### 3.13.1.1 cwp\_field\_dof\_location\_get\_()

```
integer(c_int) function cwp::cwp_field_dof_location_get::cwp_field_dof_location_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id )
```

Get target degrees of freedom location.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier

#### Returns

Location of degrees of freedom

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

- [fortran/new/cwp\\_f.f90](#)

## 3.14 cwp::cwp\_field\_interp\_function\_set Interface Reference

### Public Member Functions

- subroutine [cwp\\_field\\_interp\\_function\\_set\\_](#) (local\_code\_name, cpl\_id, field\_id, user\_interpolation\_fct)  
*Setting of a user interpolation from location.*

### 3.14.1 Member Function/Subroutine Documentation

#### 3.14.1.1 cwp\_field\_interp\_function\_set\_()

```
subroutine cwp::cwp_field_interp_function_set::cwp_field_interp_function_set_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    user_interpolation_fct )
```

Setting of a user interpolation from location.

This function takes into account an user interpolation function written with void (\*CWP\_Field\_interp\_function\_t) interface.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>fct</i>	Function

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

- [fortran/new/cwp\\_f.f90](#)

## 3.15 cwp::cwp\_field\_interp\_function\_unset Interface Reference

### Public Member Functions

- subroutine [cwp\\_field\\_interp\\_function\\_unset\\_](#) (local\_code\_name, cpl\_id, field\_id)  
*Unsetting of a user interpolation.*

### 3.15.1 Member Function/Subroutine Documentation

#### 3.15.1.1 cwp\_field\_interp\_function\_unset\_()

```
subroutine cwp::cwp_field_interp_function_unset::cwp_field_interp_function_unset_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id )
```

Unsetting of a user interpolation.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier

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

- [fortran/new/cwp\\_f.f90](#)

## 3.16 cwp::cwp\_field\_intersection\_tgt\_elt\_volumes\_get Interface Reference

### Public Member Functions

- subroutine [cwp\\_field\\_intersection\\_tgt\\_elt\\_volumes\\_get\\_](#) (local\_code\_name, cpl\_id, field\_id, i\_part, tgt\_elt↔\_volumes)  
*Get spatial local target elements volumes (intersection algorithm).*

### 3.16.1 Member Function/Subroutine Documentation

#### 3.16.1.1 cwp\_field\_intersection\_tgt\_elt\_volumes\_get\_()

```
subroutine cwp::cwp_field_intersection_tgt_elt_volumes_get::cwp_field_intersection_tgt_elt↔
volumes_get_ (
    character(kind=c_char, len = *) local_code_name,
    character(kind=c_char, len = *) cpl_id,
    character(kind=c_char, len = *) field_id,
    integer(c_int) i_part,
    double precision, dimension(:), pointer tgt_elt_volumes )
```

Get spatial local target elements volumes (intersection algorithm).

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Partition identifier
out	<i>tgt_elt_volumes</i>	Volumes of local target elements

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

- [fortran/new/cwp\\_f.f90](#)

## 3.17 cwp::cwp\_field\_intersection\_volumes\_get Interface Reference

### Public Member Functions

- subroutine [cwp\\_field\\_intersection\\_volumes\\_get\\_](#) (local\_code\_name, cpl\_id, field\_id, i\_part, volumes)  
*Get spatial interpolation volumes (intersection algorithm).*

### 3.17.1 Member Function/Subroutine Documentation

### 3.17.1.1 cwp\_field\_intersection\_volumes\_get\_()

```
subroutine cwp::cwp_field_intersection_volumes_get::cwp_field_intersection_volumes_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part,
    double precision, dimension(:), pointer volumes )
```

Get spatial interpolation volumes (intersection algorithm).

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Partition identifier
out	<i>volumes</i>	Volumes of intersection polyhedra

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

- [fortran/new/cwp\\_f.f90](#)

## 3.18 cwp::cwp\_field\_irecv Interface Reference

### Public Member Functions

- subroutine [cwp\\_field\\_irecv\\_](#) (local\_code\_name, cpl\_id, tgt\_field\_id)  
*Receive a spatially interpolated field from the coupled code with non-blocking communications.*

### 3.18.1 Member Function/Subroutine Documentation

#### 3.18.1.1 cwp\_field\_irecv\_()

```
subroutine cwp::cwp_field_irecv::cwp_field_irecv_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) tgt_field_id )
```

Receive a spatially interpolated field from the coupled code with non-blocking communications.

This function is independent of CWP\_Time\_exch\_t mode. The user has to manually check the consistency of the exchanges.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>tgt_field_id</i>	Target field id

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

- [fortran/new/cwp\\_f.f90](#)

## 3.19 cwp::cwp\_field\_issend Interface Reference

### Public Member Functions

- subroutine [cwp\\_field\\_issend\\_](#) (*local\_code\_name*, *cpl\_id*, *field\_id*)  
*Send a spatially interpolated field to the coupled code with non-blocking communications.*

### 3.19.1 Member Function/Subroutine Documentation

#### 3.19.1.1 cwp\_field\_issend\_()

```
subroutine cwp::cwp_field_issend::cwp_field_issend_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id )
```

Send a spatially interpolated field to the coupled code with non-blocking communications.

This function is independent of CWP\_Time\_exch\_t mode. The user has to manually check the consistency of the exchanges.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier

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

- [fortran/new/cwp\\_f.f90](#)

## 3.20 cwp::cwp\_field\_location\_internal\_cell\_vtx\_get Interface Reference

### Public Member Functions

- subroutine [cwp\\_field\\_location\\_internal\\_cell\\_vtx\\_get](#) (local\_code\_name, cpl\_id, field\_id, i\_part, cell\_vtx\_idx, cell\_vtx)

*Get spatial interpolation internal cell->vertex connectivity (location algorithm).*

### 3.20.1 Member Function/Subroutine Documentation

#### 3.20.1.1 cwp\_field\_location\_internal\_cell\_vtx\_get()

```
subroutine cwp::cwp_field_location_internal_cell_vtx_get::cwp_field_location_internal_cell_vtx_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part,
    integer(c_int), dimension(:), pointer cell_vtx_idx,
    integer(c_int), dimension(:), pointer cell_vtx )
```

Get spatial interpolation internal cell->vertex connectivity (location algorithm).

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Partition identifier
out	<i>cell_vtx_idx</i>	Index for local cell->vertex connectivity
out	<i>cell_vtx</i>	Local cell->vertex connectivity

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

- [fortran/new/cwp\\_f.f90](#)

## 3.21 cwp::cwp\_field\_location\_point\_data\_get Interface Reference

### Public Member Functions

- subroutine [cwp\\_field\\_location\\_point\\_data\\_get](#) (local\_code\_name, cpl\_id, field\_id, i\_part, points\_coords, points\_uvw, points\_dist2, points\_projected\_coords)

*Get spatial interpolation point data (location algorithm).*

### 3.21.1 Member Function/Subroutine Documentation

#### 3.21.1.1 cwp\_field\_location\_point\_data\_get\_()

```
subroutine cwp::cwp_field_location_point_data_get::cwp_field_location_point_data_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part,
    double precision, dimension(:,:), pointer points_coords,
    double precision, dimension(:,:), pointer points_uvw,
    double precision, dimension(:), pointer points_dist2,
    double precision, dimension(:,:), pointer points_projected_coords )
```

Get spatial interpolation point data (location algorithm).

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Partition identifier
out	<i>points_coords</i>	Cartesian coordinates of points inside local elements
out	<i>points_uvw</i>	Parametric coordinates of points inside local elements
out	<i>points_dist2</i>	Squared distance from points to elements
out	<i>points_projected_coords</i>	Cartesian coordinates of projection on points on local elements

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

- [fortran/new/cwp\\_f.f90](#)

## 3.22 cwp::cwp\_field\_location\_weights\_get Interface Reference

### Public Member Functions

- subroutine [cwp\\_field\\_location\\_weights\\_get](#) (local\_code\_name, cpl\_id, field\_id, i\_part, weights)  
*Get spatial interpolation weights (location algorithm).*

#### 3.22.1 Member Function/Subroutine Documentation



### 3.22.1.1 cwp\_field\_location\_weights\_get\_()

```
subroutine cwp::cwp_field_location_weights_get::cwp_field_location_weights_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part,
    double precision, dimension(:), pointer weights )
```

Get spatial interpolation weights (location algorithm).

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Partition identifier
out	<i>weights</i>	Interpolation weights

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

- fortran/new/[cwp\\_f.f90](#)

## 3.23 cwp::cwp\_field\_n\_components\_get Interface Reference

### Public Member Functions

- integer function [cwp\\_field\\_n\\_components\\_get\\_](#) (local\_code\_name, cpl\_id, field\_id)  
*Get spatial interpolation number of algorithms.*

### 3.23.1 Member Function/Subroutine Documentation

#### 3.23.1.1 cwp\_field\_n\_components\_get\_()

```
integer function cwp::cwp_field_n_components_get::cwp_field_n_components_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id )
```

Get spatial interpolation number of algorithms.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier

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

- [fortran/new/cwp\\_f.f90](#)

## 3.24 cwp::cwp\_field\_n\_dof\_get Interface Reference

### Public Member Functions

- integer(c\_int) function [cwp\\_field\\_n\\_dof\\_get\\_](#) (local\_code\_name, cpl\_id, field\_id, i\_part)  
*Get field number of degrees of freedom.*

### 3.24.1 Member Function/Subroutine Documentation

#### 3.24.1.1 cwp\_field\_n\_dof\_get\_()

```
integer(c_int) function cwp::cwp_field_n_dof_get::cwp_field_n_dof_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part )
```

Get field number of degrees of freedom.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>n_dof</i>	Field number of degrees of freedom

#### Returns

Field storage type

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

- [fortran/new/cwp\\_f.f90](#)

## 3.25 cwp::cwp\_field\_nearest\_neighbors\_coord\_get Interface Reference

### Public Member Functions

- subroutine [cwp\\_field\\_nearest\\_neighbors\\_coord\\_get\\_](#) (local\_code\_name, cpl\_id, field\_id, i\_part, nearest\_↔  
src\_coord)  
*Get coordinates of nearest source points (nearest neighbors algorithm).*

### 3.25.1 Member Function/Subroutine Documentation

#### 3.25.1.1 cwp\_field\_nearest\_neighbors\_coord\_get\_()

```
subroutine cwp::cwp_field_nearest_neighbors_coord_get::cwp_field_nearest_neighbors_coord_get←
_ (
    character(kind=c_char, len = *) local_code_name,
    character(kind=c_char, len = *) cpl_id,
    character(kind=c_char, len = *) field_id,
    integer(c_int), intent(in) i_part,
    double precision, dimension(:,:), pointer nearest_src_coord )
```

Get coordinates of nearest source points (nearest neighbors algorithm).

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Partition identifier
out	<i>nearest_src_coord</i>	Coordinates of nearest source points

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

- fortran/new/[cwp\\_f.f90](#)

## 3.26 cwp::cwp\_field\_nearest\_neighbors\_distances\_get Interface Reference

### Public Member Functions

- subroutine [cwp\\_field\\_nearest\\_neighbors\\_distances\\_get\\_](#) (local\_code\_name, cpl\_id, field\_id, i\_part, distances2)

*Get spatial interpolation distances (nearest neighbors algorithm).*

### 3.26.1 Member Function/Subroutine Documentation

#### 3.26.1.1 cwp\_field\_nearest\_neighbors\_distances\_get\_()

```
subroutine cwp::cwp_field_nearest_neighbors_distances_get::cwp_field_nearest_neighbors←
distances_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part,
    double precision, dimension(:), pointer distances2 )
```

Get spatial interpolation distances (nearest neighbors algorithm).

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Partition identifier
out	<i>distances2</i>	Squared distances from nearest source points

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

- [fortran/new/cwp\\_f.f90](#)

## 3.27 cwp::cwp\_field\_src\_data\_properties\_get Interface Reference

### Public Member Functions

- subroutine [cwp\\_field\\_src\\_data\\_properties\\_get\\_](#) (*local\_code\_name*, *cpl\_id*, *field\_id*, *i\_part*, *n\_elt\_src*, *src\_to\_tgt\_idx*)  
*Get spatial interpolation source data.*

### 3.27.1 Member Function/Subroutine Documentation

#### 3.27.1.1 cwp\_field\_src\_data\_properties\_get\_()

```
subroutine cwp::cwp_field_src_data_properties_get::cwp_field_src_data_properties_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part,
    integer(c_int) n_elt_src,
    integer(c_int), dimension(:), pointer src_to_tgt_idx )
```

Get spatial interpolation source data.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
out	<i>i_part</i>	Partition identifier
out	<i>n_elt_src</i>	Number of local source entities in current partition
out	<i>src_to_tgt_idx</i>	Index for source->target mapping

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

- [fortran/new/cwp\\_f.f90](#)

## 3.28 cwp::cwp\_field\_storage\_get Interface Reference

### Public Member Functions

- integer(c\_int) function [cwp\\_field\\_storage\\_get\\_](#) (local\_code\_name, cpl\_id, field\_id)  
*Get field storage type.*

### 3.28.1 Member Function/Subroutine Documentation

#### 3.28.1.1 cwp\_field\_storage\_get\_()

```
integer(c_int) function cwp::cwp_field_storage_get::cwp_field_storage_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id )
```

Get field storage type.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier

#### Returns

Field storage type

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

- [fortran/new/cwp\\_f.f90](#)

## 3.29 cwp::cwp\_field\_tgt\_data\_properties\_get Interface Reference

### Public Member Functions

- subroutine [cwp\\_field\\_tgt\\_data\\_properties\\_get\\_](#) (local\_code\_name, cpl\_id, field\_id, i\_part, n\_elt\_tgt, n\_↵referenced\_tgt, referenced\_tgt, tgt\_come\_from\_src\_idx)  
*Get spatial interpolation target data.*

### 3.29.1 Member Function/Subroutine Documentation

#### 3.29.1.1 cwp\_field\_tgt\_data\_properties\_get\_()

```
subroutine cwp::cwp_field_tgt_data_properties_get::cwp_field_tgt_data_properties_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part,
    integer(c_int) n_elt_tgt,
    integer(c_int) n_referenced_tgt,
    integer(c_int), dimension(:), pointer referenced_tgt,
    integer(c_int), dimension(:), pointer tgt_come_from_src_idx )
```

Get spatial interpolation target data.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
out	<i>i_part</i>	Partition identifier
out	<i>n_elt_tgt</i>	Number of local target entities in current partition
out	<i>n_referenced_tgt</i>	Number of referenced target entities in current partition
out	<i>referenced_tgt</i>	Ids of referenced target entities in current partition (1-based)
out	<i>tgt_come_from_src_idx</i>	Index for target->source mapping

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

- [fortran/new/cwp\\_f.f90](#)

## 3.30 cwp::cwp\_field\_wait\_irecv Interface Reference

### Public Member Functions

- subroutine [cwp\\_field\\_wait\\_irecv\\_](#) (local\_code\_name, cpl\_id, tgt\_field\_id)  
Wait the end of an exchange related to request from CWP\_Field\_irecv.

#### 3.30.1 Member Function/Subroutine Documentation

### 3.30.1.1 cwp\_field\_wait\_irecv\_()

```
subroutine cwp::cwp_field_wait_irecv::cwp_field_wait_irecv_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) tgt_field_id )
```

Wait the end of an exchange related to request from CWP\_Field\_irecv.

This function waits the end of exchange related to request from CWP\_Field\_irecv

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>tgt_field_id</i>	Target field id

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

- [fortran/new/cwp\\_f.f90](#)

## 3.31 cwp::cwp\_field\_wait\_issend Interface Reference

### Public Member Functions

- subroutine [cwp\\_field\\_wait\\_issend\\_](#) (local\_code\_name, cpl\_id, field\_id)  
*Wait the end of an exchange related to request from CWP\_Field\_issend.*

### 3.31.1 Member Function/Subroutine Documentation

#### 3.31.1.1 cwp\_field\_wait\_issend\_()

```
subroutine cwp::cwp_field_wait_issend::cwp_field_wait_issend_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id )
```

Wait the end of an exchange related to request from CWP\_Field\_issend.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier

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

- [fortran/new/cwp\\_f.f90](#)

## 3.32 cwp::CWP\_Finalize Interface Reference

Finalize CWIPI.

### Public Member Functions

- subroutine **cwp\_finalize** ()

### 3.32.1 Detailed Description

Finalize CWIPI.

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

- [fortran/new/cwp\\_f.f90](#)

## 3.33 cwp::cwp\_global\_data\_irecv Interface Reference

### Public Member Functions

- subroutine [cwp\\_global\\_data\\_irecv\\_int](#) (local\_code\_name, cpl\_id, global\_data\_id, recv\_data)  
*Initiate the reception of a data array.*
- subroutine **cwp\_global\_data\_irecv\_long** (local\_code\_name, cpl\_id, global\_data\_id, recv\_data)
- subroutine **cwp\_global\_data\_irecv\_double** (local\_code\_name, cpl\_id, global\_data\_id, recv\_data)
- subroutine **cwp\_global\_data\_irecv\_complex4** (local\_code\_name, cpl\_id, global\_data\_id, recv\_data)
- subroutine **cwp\_global\_data\_irecv\_complex8** (local\_code\_name, cpl\_id, global\_data\_id, recv\_data)
- subroutine **cwp\_global\_data\_irecv\_real4** (local\_code\_name, cpl\_id, global\_data\_id, recv\_data)

### 3.33.1 Member Function/Subroutine Documentation

#### 3.33.1.1 cwp\_global\_data\_irecv\_int()

```
subroutine cwp::cwp_global_data_irecv::cwp_global_data_irecv_int (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id,
    character(kind=c_char, len=*) global_data_id,
    integer(c_int), dimension(:, :), pointer recv_data )
```

Initiate the reception of a data array.



## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>global_data_id</i>	GlobalData identifier
in	<i>recv_data</i>	Pointer to data array

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

- [fortran/new/cwp\\_f.f90](#)

## 3.34 cwp::cwp\_global\_data\_issend Interface Reference

### Public Member Functions

- subroutine [cwp\\_global\\_data\\_issend\\_int](#) (local\_code\_name, cpl\_id, global\_data\_id, send\_data)  
*Initiate the sending of a data array.*
- subroutine **cwp\_global\_data\_issend\_long** (local\_code\_name, cpl\_id, global\_data\_id, send\_data)
- subroutine **cwp\_global\_data\_issend\_double** (local\_code\_name, cpl\_id, global\_data\_id, send\_data)
- subroutine **cwp\_global\_data\_issend\_complex4** (local\_code\_name, cpl\_id, global\_data\_id, send\_data)
- subroutine **cwp\_global\_data\_issend\_complex8** (local\_code\_name, cpl\_id, global\_data\_id, send\_data)
- subroutine **cwp\_global\_data\_issend\_real4** (local\_code\_name, cpl\_id, global\_data\_id, send\_data)

### 3.34.1 Member Function/Subroutine Documentation

#### 3.34.1.1 cwp\_global\_data\_issend\_int()

```
subroutine cwp::cwp_global_data_issend::cwp_global_data_issend_int (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id,
    character(kind=c_char, len=*) global_data_id,
    integer(c_int), dimension(:, :), pointer send_data )
```

Initiate the sending of a data array.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>global_data_id</i>	GlobalData identifier
in	<i>send_data</i>	Pointer to data array

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

- [fortran/new/cwp\\_f.f90](#)

## 3.35 cwp::cwp\_global\_data\_wait\_irecv Interface Reference

### Public Member Functions

- subroutine [cwp\\_global\\_data\\_wait\\_irecv\\_](#) (local\_code\_name, cpl\_id, global\_data\_id)  
*Finalize the reception of a data array.*

### 3.35.1 Member Function/Subroutine Documentation

#### 3.35.1.1 cwp\_global\_data\_wait\_irecv\_()

```
subroutine cwp::cwp_global_data_wait_irecv::cwp_global_data_wait_irecv_ (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id,
    character(kind=c_char, len=*) global_data_id )
```

Finalize the reception of a data array.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>global_data_id</i>	GlobalData identifier

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

- [fortran/new/cwp\\_f.f90](#)

## 3.36 cwp::cwp\_global\_data\_wait\_issend Interface Reference

### Public Member Functions

- subroutine [cwp\\_global\\_data\\_wait\\_issend\\_](#) (local\_code\_name, cpl\_id, global\_data\_id)  
*Finalize the sending of a data array.*

### 3.36.1 Member Function/Subroutine Documentation

#### 3.36.1.1 cwp\_global\_data\_wait\_issend\_()

```
subroutine cwp::cwp_global_data_wait_issend::cwp_global_data_wait_issend_ (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id,
    character(kind=c_char, len=*) global_data_id )
```

Finalize the sending of a data array.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>global_data_id</i>	GlobalData identifier

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

- fortran/new/[cwp\\_f.f90](#)

## 3.37 cwp::cwp\_init Interface Reference

### Public Member Functions

- subroutine [cwp\\_init](#) (fcomm, n\_code, code\_names, is\_active\_rank, intra\_comms)  
*Initialize CWIPI.*

### 3.37.1 Member Function/Subroutine Documentation

#### 3.37.1.1 cwp\_init\_()

```
subroutine cwp::cwp_init::cwp_init_ (
    integer(c_int) fcomm,
    integer(c_int), intent(in) n_code,
    character(kind = c_char, len = *), dimension(n_code), target code_names,
    integer(c_int) is_active_rank,
    integer(c_int), dimension(:), pointer intra_comms )
```

Initialize CWIPI.

This function creates the MPI intra communicators of the codes from the `global_comm` MPI communicator that contains all code ranks. This function has to be called from all ranks contained in the `global_comm`.

## Parameters

in	<i>global_comm</i>	MPI global communicator
in	<i>n_code</i>	Number of codes on the current rank
in	<i>code_names</i>	Names of codes on the current rank (size = <code>n_code</code> )
in	<i>is_active_rank</i>	Current rank is available for CWIPI
out	<i>intra_comms</i>	MPI intra communicators of each code (size = <code>n_code</code> )

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

- fortran/new/[cwp\\_f.f90](#)

### 3.38 cwp::cwp\_involved\_srcs\_bcast\_enable Interface Reference

#### Public Member Functions

- subroutine [cwp\\_involved\\_srcs\\_bcast\\_enable\\_](#) (local\_code\_name, cpl\_id, field\_id)  
*Enable broadcast of the involved sources ids (in CWP\_COMM\_PAR\_WITHOUT\_PART mode).*

#### 3.38.1 Member Function/Subroutine Documentation

##### 3.38.1.1 cwp\_involved\_srcs\_bcast\_enable\_()

```
subroutine cwp::cwp_involved_srcs_bcast_enable::cwp_involved_srcs_bcast_enable_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id )
```

Enable broadcast of the involved sources ids (in CWP\_COMM\_PAR\_WITHOUT\_PART mode).

This function must be called in order for the involved sources to be accessible on non-root ranks

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Current partition

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

- [fortran/new/cwp\\_f.f90](#)

### 3.39 cwp::cwp\_involved\_srcs\_get Interface Reference

#### Public Member Functions

- integer(c\_int) function, dimension(:), pointer [cwp\\_involved\\_srcs\\_get\\_](#) (local\_code\_name, cpl\_id, field\_id, i↔\_part)  
*Return involved sources.*

#### 3.39.1 Member Function/Subroutine Documentation

### 3.39.1.1 cwp\_involved\_srcs\_get\_()

```
integer(c_int) function, dimension(:), pointer cwp::cwp_involved_srcs_get::cwp_involved_srcs←
_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part )
```

Return involved sources.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Current partition

#### Returns

Involved sources

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

- [fortran/new/cwp\\_f.f90](#)

## 3.40 cwp::cwp\_loc\_codes\_list\_get Interface Reference

### Public Member Functions

- `character(256) function, dimension(:), allocatable cwp::cwp_loc_codes_list_get::cwp_loc_codes_list_get_()`  
*Return list of local codes known by CWIPI.*

### 3.40.1 Member Function/Subroutine Documentation

#### 3.40.1.1 cwp\_loc\_codes\_list\_get\_()

```
character(256) function, dimension(:), allocatable cwp::cwp_loc_codes_list_get::cwp_loc_codes_list_get_
```

Return list of local codes known by CWIPI.

#### Returns

List of local code names

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

- [fortran/new/cwp\\_f.f90](#)

### 3.41 cwp::CWP\_Loc\_codes\_nb\_get Interface Reference

Return the number of local codes known by CWIPI.

#### Public Member Functions

- integer(c\_int) function **cwp\_loc\_codes\_nb\_get** ()

#### 3.41.1 Detailed Description

Return the number of local codes known by CWIPI.

##### Returns

Number of local codes

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

- fortran/new/[cwp\\_f.f90](#)

### 3.42 cwp::cwp\_mesh\_interf\_block\_add Interface Reference

#### Public Member Functions

- integer(c\_int) function [cwp\\_mesh\\_interf\\_block\\_add](#) (local\_code\_name, cpl\_id, block\_type)  
*Add a connectivity block to the interface mesh.*

#### 3.42.1 Member Function/Subroutine Documentation

##### 3.42.1.1 cwp\_mesh\_interf\_block\_add\_()

```
integer(c_int) function cwp::cwp_mesh_interf_block_add::cwp_mesh_interf_block_add_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    integer(c_int) block_type )
```

Add a connectivity block to the interface mesh.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>block_type</i>	Block type

**Returns**

block identifier

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

- [fortran/new/cwp\\_f.f90](#)

## 3.43 cwp::cwp\_mesh\_interf\_block\_std\_get Interface Reference

### Public Member Functions

- subroutine [cwp\\_mesh\\_interf\\_block\\_std\\_get\\_](#) (local\_code\_name, cpl\_id, i\_part, block\_id, n\_elts, connec, global\_num)

*Get the properties of a standard block of the interface mesh.*

### 3.43.1 Member Function/Subroutine Documentation

#### 3.43.1.1 cwp\_mesh\_interf\_block\_std\_get\_()

```
subroutine cwp::cwp_mesh_interf_block_std_get::cwp_mesh_interf_block_std_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    integer(c_int) i_part,
    integer(c_int) block_id,
    integer(c_int) n_elts,
    integer(c_int), dimension(:), pointer connec,
    integer(c_long), dimension(:), pointer global_num )
```

Get the properties of a standard block of the interface mesh.

**Parameters**

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Partition identifier
in	<i>block_id</i>	Block identifier
out	<i>n_elts</i>	Number of elements
out	<i>connec</i>	Connectivity (size = n_vertex_elt * n_elts)
out	<i>global_num</i>	Pointer to global element number (or NULL)

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

- [fortran/new/cwp\\_f.f90](#)

## 3.44 cwp::cwp\_mesh\_intf\_block\_std\_set Interface Reference

### Public Member Functions

- subroutine [cwp\\_mesh\\_intf\\_block\\_std\\_set\\_](#) (local\_code\_name, cpl\_id, i\_part, block\_id, n\_elts, connec, global\_num)

*Set a standard block to the interface mesh.*

### 3.44.1 Member Function/Subroutine Documentation

#### 3.44.1.1 cwp\_mesh\_intf\_block\_std\_set\_()

```
subroutine cwp::cwp_mesh_intf_block_std_set::cwp_mesh_intf_block_std_set_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    integer(c_int) i_part,
    integer(c_int) block_id,
    integer(c_int) n_elts,
    integer(c_int), dimension(:), pointer connec,
    integer(c_long), dimension(:), pointer global_num )
```

Set a standard block to the interface mesh.

This function adds a connectivity block to the interface mesh. Definition of element connectivity is :

- edge (CWP\_BLOCK\_EDGE2) :

```
1 x-----x 2
```

- triangle (CWP\_BLOCK\_FACE\_TRIA3):

```
1 x-----x 3
   \      /
    \    /
     \  /
      x 2
```

- quadrangle (CWP\_BLOCK\_FACE\_QUAD4) :

```
4 x-----x 3
 /      \
/        \
/          \
1 x-----x 2
```

- tetrahedron (CWP\_BLOCK\_CELL\_TETRA4) :

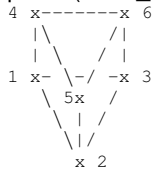
```
      x 4
     / \
    /   \
   /     \
  /       \
 /         \
1 x-----x 3
   \       /
    \     /
     \   /
      x 2
```

- pyramid (CWP\_BLOCK\_CELL\_PYRAM5) :

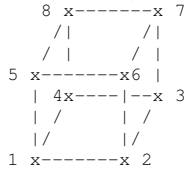
```
      5 x
     /  \
    /    \
   /      \
  /        \
 /          \
4 x-----x 3
 /          \
/            \
/              \
1 x-----x 2
```



- prism (CWP\_BLOCK\_CELL\_PRISM6) :



- hexaedron (CWP\_BLOCK\_CELL\_HEX8) :



#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Partition identifier
in	<i>block_id</i>	Block identifier
in	<i>n_elts</i>	Number of elements
in	<i>connec</i>	Connectivity (size = <i>n_vertex_elt</i> * <i>n_elts</i> )
in	<i>global_num</i>	Pointer to global element number (or NULL)

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

- [fortran/new/cwp\\_f.f90](#)

## 3.45 cwp::cwp\_mesh\_intf\_c\_poly\_block\_get Interface Reference

### Public Member Functions

- subroutine [cwp\\_mesh\\_intf\\_c\\_poly\\_block\\_get\\_](#) (*local\_code\_name*, *cpl\_id*, *i\_part*, *block\_id*, *n\_elts*, *n\_faces*, *connec\_faces\_idx*, *connec\_faces*, *connec\_cells\_idx*, *connec\_cells*, *global\_num*)

*Get the properties of a polyhedron block of the interface mesh partition..*

### 3.45.1 Member Function/Subroutine Documentation

#### 3.45.1.1 cwp\_mesh\_intf\_c\_poly\_block\_get\_()

```
subroutine cwp::cwp_mesh_intf_c_poly_block_get::cwp_mesh_intf_c_poly_block_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    integer(c_int) i_part,
    integer(c_int) block_id,
    integer(c_int) n_elts,
```

```

integer(c_int) n_faces,
integer(c_int), dimension(:), pointer connec_faces_idx,
integer(c_int), dimension(:), pointer connec_faces,
integer(c_int), dimension(:), pointer connec_cells_idx,
integer(c_int), dimension(:), pointer connec_cells,
integer(c_long), dimension(:), pointer global_num )

```

Get the properties of a polyhedron block of the interface mesh partition..

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Current partition
in	<i>block_id</i>	Block identifier
out	<i>n_elts</i>	Number of elements
out	<i>connec_cells_idx</i>	Polyhedron to face index ( <i>connec_cells_idx</i> [0] = 0 and size = <i>n_elts</i> + 1)
out	<i>connec_cells</i>	Polyhedron to face connectivity (size = <i>connec_cells_idx</i> [ <i>n_elts</i> ])
out	<i>n_faces</i>	Number of faces
out	<i>connec_faces_idx</i>	Polyhedron face to vertex index ( <i>connec_faces_idx</i> [0] = 0 and size = max( <i>cell_face_connec</i> ) + 1)
out	<i>connec_faces</i>	Polyhedron face to vertex connectivity (size = <i>connec_faces_idx</i> [ <i>n_elts</i> ])
out	<i>global_num</i>	Pointer to global element number (or NULL)

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

- [fortran/new/cwp\\_f.f90](#)

## 3.46 cwp::cwp\_mesh\_interf\_c\_poly\_block\_set Interface Reference

### Public Member Functions

- subroutine [cwp\\_mesh\\_interf\\_c\\_poly\\_block\\_set\\_](#) (*local\_code\_name*, *cpl\_id*, *i\_part*, *block\_id*, *n\_elts*, *n\_faces*, *connec\_faces\_idx*, *connec\_faces*, *connec\_cells\_idx*, *connec\_cells*, *global\_num*)

*Adding a polyhedron connectivity block to the interface mesh.*

### 3.46.1 Member Function/Subroutine Documentation

### 3.46.1.1 cwp\_mesh\_interf\_c\_poly\_block\_set\_()

```
subroutine cwp::cwp_mesh_interf_c_poly_block_set::cwp_mesh_interf_c_poly_block_set_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    integer(c_int) i_part,
    integer(c_int) block_id,
    integer(c_int) n_elts,
    integer(c_int) n_faces,
    integer(c_int), dimension(:), pointer connec_faces_idx,
    integer(c_int), dimension(:), pointer connec_faces,
    integer(c_int), dimension(:), pointer connec_cells_idx,
    integer(c_int), dimension(:), pointer connec_cells,
    integer(c_long), dimension(:), pointer global_num )
```

Adding a polyhedron connectivity block to the interface mesh.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Current partition
in	<i>block_id</i>	Block identifier
in	<i>n_elts</i>	Number of elements
in	<i>connec_cells_idx</i>	Polyhedron to face index ( <i>src_poly_cell_face_idx</i> [0] = 0 and size = <i>n_elts</i> + 1)
in	<i>connec_cells</i>	Polyhedron to face connectivity (size = <i>cell_face_idx</i> [ <i>n_elts</i> ])
in	<i>n_faces</i>	Number of faces
in	<i>connec_faces_idx</i>	Polyhedron face to vertex index ( <i>connec_faces_idx</i> [0] = 0 and size = <i>max</i> ( <i>cell_face_connec</i> ) + 1)
in	<i>connec_faces</i>	Polyhedron face to vertex connectivity (size = <i>connec_faces_idx</i> [ <i>n_elts</i> ])
in	<i>global_num</i>	Pointer to global element number (or NULL)

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

- [fortran/new/cwp\\_f.f90](#)

## 3.47 cwp::cwp\_mesh\_interf\_del Interface Reference

### Public Member Functions

- subroutine [cwp\\_mesh\\_interf\\_del\\_](#) (*local\_code\_name*, *cpl\_id*)  
*Delete interface mesh.*

### 3.47.1 Member Function/Subroutine Documentation

### 3.47.1.1 cwp\_mesh\_interf\_del\_()

```
subroutine cwp::cwp_mesh_interf_del::cwp_mesh_interf_del_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id )
```

Delete interface mesh.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier

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

- [fortran/new/cwp\\_f.f90](#)

## 3.48 cwp::cwp\_mesh\_interf\_f\_poly\_block\_get Interface Reference

### Public Member Functions

- subroutine [cwp\\_mesh\\_interf\\_f\\_poly\\_block\\_get\\_](#) (*local\_code\_name*, *cpl\_id*, *i\_part*, *block\_id*, *n\_elts*, *connec*↵  
\_idx, *connec*, *global\_num*)  
*Get the properties of a polygon block of the interface mesh partition.*

### 3.48.1 Member Function/Subroutine Documentation

#### 3.48.1.1 cwp\_mesh\_interf\_f\_poly\_block\_get\_()

```
subroutine cwp::cwp_mesh_interf_f_poly_block_get::cwp_mesh_interf_f_poly_block_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    integer(c_int) i_part,
    integer(c_int) block_id,
    integer(c_int) n_elts,
    integer(c_int), dimension(:), pointer connec_idx,
    integer(c_int), dimension(:), pointer connec,
    integer(c_long), dimension(:), pointer global_num )
```

Get the properties of a polygon block of the interface mesh partition.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Current partition
in	<i>block_id</i>	Block identifier
out	<i>n_elts</i>	Number of elements
out	<i>connec_idx</i>	Connectivity index ( <i>connec_idx</i> [0] = 0 and size = <i>n_elts</i> + 1)
out	<i>connec</i>	Connectivity (size = <i>connec_idx</i> [ <i>n_elts</i> ])
out	<i>global_num</i>	Pointer to global element number (or NULL)

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

- [fortran/new/cwp\\_f.f90](#)

## 3.49 cwp::cwp\_mesh\_intf\_f\_poly\_block\_set Interface Reference

### Public Member Functions

- subroutine [cwp\\_mesh\\_intf\\_f\\_poly\\_block\\_set\\_](#) (local\_code\_name, cpl\_id, i\_part, block\_id, n\_elts, connec←\_idx, connec, global\_num)

*Set the connectivity of a polygon block in a interface mesh partition.*

### 3.49.1 Member Function/Subroutine Documentation

#### 3.49.1.1 cwp\_mesh\_intf\_f\_poly\_block\_set\_()

```
subroutine cwp::cwp_mesh_intf_f_poly_block_set::cwp_mesh_intf_f_poly_block_set_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    integer(c_int) i_part,
    integer(c_int) block_id,
    integer(c_int) n_elts,
    integer(c_int), dimension(:), pointer connec_idx,
    integer(c_int), dimension(:), pointer connec,
    integer(c_long), dimension(:), pointer global_num )
```

Set the connectivity of a polygon block in a interface mesh partition.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Current partition
in	<i>block_id</i>	Block identifier
in	<i>n_elts</i>	Number of elements
in	<i>connec_idx</i>	Connectivity index ( <i>connec_idx</i> [0] = 0 and size = <i>n_elts</i> + 1)
in	<i>connec</i>	Connectivity (size = <i>connec_idx</i> [ <i>n_elts</i> ])
in	<i>global_num</i>	Pointer to global element number (or NULL)

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

- [fortran/new/cwp\\_f.f90](#)

## 3.50 cwp::cwp\_mesh\_interf\_finalize Interface Reference

### Public Member Functions

- subroutine [cwp\\_mesh\\_interf\\_finalize\\_](#) (local\_code\_name, cpl\_id)  
*Finalize interface mesh.*

### 3.50.1 Member Function/Subroutine Documentation

#### 3.50.1.1 cwp\_mesh\_interf\_finalize\_()

```
subroutine cwp::cwp_mesh_interf_finalize::cwp_mesh_interf_finalize_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id )
```

Finalize interface mesh.

This function computes the global numbers of mesh entities if they are not provided.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier

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

- [fortran/new/cwp\\_f.f90](#)

## 3.51 cwp::cwp\_mesh\_interf\_from\_cellface\_set Interface Reference

### Public Member Functions

- subroutine [cwp\\_mesh\\_interf\\_from\\_cellface\\_set\\_](#) (local\_code\_name, cpl\_id, i\_part, n\_cells, cell\_face\_idx, cell\_face, n\_faces, face\_vtx\_idx, face\_vtx, global\_num)  
*Define the interface mesh from a cell to face connectivity.*

### 3.51.1 Member Function/Subroutine Documentation

### 3.51.1.1 cwp\_mesh\_intf\_from\_cellface\_set\_()

```

subroutine cwp::cwp_mesh_intf_from_cellface_set::cwp_mesh_intf_from_cellface_set_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    integer(c_int) i_part,
    integer(c_int) n_cells,
    integer(c_int), dimension(:), pointer cell_face_idx,
    integer(c_int), dimension(:), pointer cell_face,
    integer(c_int) n_faces,
    integer(c_int), dimension(:), pointer face_vtx_idx,
    integer(c_int), dimension(:), pointer face_vtx,
    integer(c_long), dimension(:), pointer global_num )

```

Define the interface mesh from a cell to face connectivity.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Current partition
in	<i>n_cells</i>	Number of cells
in	<i>cell_face_idx</i>	Polyhedron to face index ( <i>src_poly_cell_face_idx</i> [0] = 0 and size = <i>n_elts</i> + 1)
in	<i>cell_face</i>	Cell to face connectivity (size = <i>cell_face_idx</i> [ <i>n_elts</i> ])
in	<i>n_faces</i>	Number of faces
in	<i>face_vtx_idx</i>	Polyhedron face to vertex index ( <i>face_vtx_idx</i> [0] = 0 and size = <i>n_faces</i> + 1)
in	<i>face_vtx</i>	Face to vertex connectivity (size = <i>face_vtx_idx</i> [ <i>n_elts</i> ])
in	<i>global_num</i>	Pointer to parent element number (or NULL)

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

- [fortran/new/cwp\\_f.f90](#)

## 3.52 cwp::cwp\_mesh\_intf\_from\_faceedge\_set Interface Reference

### Public Member Functions

- subroutine [cwp\\_mesh\\_intf\\_from\\_faceedge\\_set\\_](#) (*local\_code\_name*, *cpl\_id*, *i\_part*, *n\_faces*, *face\_edge\_idx*, *face\_edge*, *n\_edges*, *edge\_vtx*, *global\_num*)  
*Define the surface interface mesh from a face to edge connectivity.*

### 3.52.1 Member Function/Subroutine Documentation

### 3.52.1.1 cwp\_mesh\_interf\_from\_faceedge\_set\_()

```
subroutine cwp::cwp_mesh_interf_from_faceedge_set::cwp_mesh_interf_from_faceedge_set_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    integer(c_int) i_part,
    integer(c_int) n_faces,
    integer(c_int), dimension(:), pointer face_edge_idx,
    integer(c_int), dimension(:), pointer face_edge,
    integer(c_int) n_edges,
    integer(c_int), dimension(:), pointer edge_vtx,
    integer(c_long), dimension(:), pointer global_num )
```

Define the surface interface mesh from a face to edge connectivity.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Current partition
in	<i>n_faces</i>	Number of cells
in	<i>face_edge_idx</i>	Polygon to edge index ( <i>face_edge_idx</i> [0] = 0 and size = <i>n_faces</i> + 1)
in	<i>face_edge</i>	Face to edge connectivity (size = <i>face_edge_idx</i> [ <i>n_faces</i> ])
in	<i>n_edges</i>	Number of faces
in	<i>edge_vtx</i>	Edge to vertex connectivity (size = 2 * <i>n_edges</i> )
in	<i>global_num</i>	Pointer to parent element number (or NULL)

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

- fortran/new/[cwp\\_f.f90](#)

## 3.53 cwp::cwp\_mesh\_interf\_vtx\_set Interface Reference

### Public Member Functions

- subroutine [cwp\\_mesh\\_interf\\_vtx\\_set](#) (local\_code\_name, cpl\_id, i\_part, n\_pts, coord, global\_num)  
*Set vertices.*

### 3.53.1 Member Function/Subroutine Documentation

#### 3.53.1.1 cwp\_mesh\_interf\_vtx\_set\_()

```
subroutine cwp::cwp_mesh_interf_vtx_set::cwp_mesh_interf_vtx_set_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    integer(c_int) i_part,
    integer(c_int) n_pts,
    double precision, dimension(:,:), pointer coord,
    integer(c_long), dimension(:), pointer global_num )
```

Set vertices.



## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Current partition
in	<i>n_pts</i>	Number of points
in	<i>coord</i>	Coordinates (size = 3 * <i>n_pts</i> )
in	<i>global_num</i>	Pointer to parent element number (or NULL)

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

- [fortran/new/cwp\\_f.f90](#)

## 3.54 cwp::cwp\_n\_computed\_tgts\_get Interface Reference

### Public Member Functions

- integer(c\_int) function [cwp\\_n\\_computed\\_tgts\\_get\\_](#) (*local\_code\_name*, *cpl\_id*, *field\_id*, *i\_part*)  
*Return the number of computed targets.*

### 3.54.1 Member Function/Subroutine Documentation

#### 3.54.1.1 cwp\_n\_computed\_tgts\_get\_()

```
integer(c_int) function cwp::cwp_n_computed_tgts_get::cwp_n_computed_tgts_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part )
```

Return the number of computed targets.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Current partition

## Returns

Number of computed targets

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

- [fortran/new/cwp\\_f.f90](#)

## 3.55 cwp::cwp\_n\_involved\_srcs\_get Interface Reference

### Public Member Functions

- integer(c\_int) function [cwp\\_n\\_involved\\_srcs\\_get\\_](#) (local\_code\_name, cpl\_id, field\_id, i\_part)  
*Return the number of involved sources.*

### 3.55.1 Member Function/Subroutine Documentation

#### 3.55.1.1 cwp\_n\_involved\_srcs\_get\_()

```
integer(c_int) function cwp::cwp_n_involved_srcs_get::cwp_n_involved_srcs_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part )
```

Return the number of involved sources.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Current partition

#### Returns

Number of involved sources

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

- [fortran/new/cwp\\_f.f90](#)

## 3.56 cwp::cwp\_n\_uncomputed\_tgts\_get Interface Reference

### Public Member Functions

- integer(c\_int) function [cwp\\_n\\_uncomputed\\_tgts\\_get\\_](#) (local\_code\_name, cpl\_id, field\_id, i\_part)  
*Return the number of uncomputed targets.*

### 3.56.1 Member Function/Subroutine Documentation

#### 3.56.1.1 cwp\_n\_uncomputed\_tgts\_get\_()

```
integer(c_int) function cwp::cwp_n_uncomputed_tgts_get::cwp_n_uncomputed_tgts_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part )
```

Return the number of uncomputed targets.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Current partition

##### Returns

Number of uncomputed targets

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

- [fortran/new/cwp\\_f.f90](#)

## 3.57 cwp::cwp\_output\_file\_set Interface Reference

### Public Member Functions

- subroutine [cwp\\_output\\_file\\_set\\_](#) (f\_output\_file\_name)  
*Define output file (in which only C code writes).*

### 3.57.1 Member Function/Subroutine Documentation

#### 3.57.1.1 cwp\_output\_file\_set\_()

```
subroutine cwp::cwp_output_file_set::cwp_output_file_set_ (
    character(kind = c_char, len = *) f_output_file_name )
```

Define output file (in which only C code writes).

## Parameters

in	<i>output_file_name</i>	Output file name
----	-------------------------	------------------

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

- [fortran/new/cwp\\_f.f90](#)

## 3.58 cwp::cwp\_param\_add Interface Reference

### Public Member Functions

- subroutine [cwp\\_param\\_add\\_int\\_](#) (local\_code\_name, param\_name, initial\_value)  
*Add a new parameter and initialize it.*
- subroutine [cwp\\_param\\_add\\_double\\_](#) (local\_code\_name, param\_name, initial\_value)
- subroutine [cwp\\_param\\_add\\_char\\_](#) (local\_code\_name, param\_name, initial\_value)

### 3.58.1 Member Function/Subroutine Documentation

#### 3.58.1.1 cwp\_param\_add\_int\_()

```
subroutine cwp::cwp_param_add::cwp_param_add_int_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) param_name,
    integer(kind = c_int), intent(in) initial_value )
```

Add a new parameter and initialize it.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>param_name</i>	Parameter name
in	<i>data_type</i>	Parameter type
in	<i>initial_value</i>	Initial value

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

- [fortran/new/cwp\\_f.f90](#)

## 3.59 cwp::cwp\_param\_del Interface Reference

### Public Member Functions

- subroutine [cwp\\_param\\_del\\_](#) (local\_code\_name, param\_name, data\_type)

*Delete a parameter.*

### 3.59.1 Member Function/Subroutine Documentation

#### 3.59.1.1 cwp\_param\_del\_()

```
subroutine cwp::cwp_param_del::cwp_param_del_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) param_name,
    integer, intent(in) data_type )
```

Delete a parameter.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>param_name</i>	Parameter name
in	<i>data_type</i>	Parameter type,

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

- [fortran/new/cwp\\_f.f90](#)

## 3.60 cwp::cwp\_param\_get Interface Reference

### Public Member Functions

- subroutine [cwp\\_param\\_get\\_int](#) (code\_name, param\_name, value)  
*Return the parameter value of param\_name on code\_name.*
- subroutine [cwp\\_param\\_get\\_double](#) (code\_name, param\_name, value)
- subroutine [cwp\\_param\\_get\\_char](#) (code\_name, param\_name, val)

### 3.60.1 Member Function/Subroutine Documentation

#### 3.60.1.1 cwp\_param\_get\_int()

```
subroutine cwp::cwp_param_get::cwp_param_get_int (
    character(kind = c_char, len = *) code_name,
    character(kind = c_char, len = *) param_name,
    integer(c_int), intent(out) value )
```

Return the parameter value of param\_name on code\_name.

**Parameters**

in	<i>code_name</i>	Local or distant code name
in	<i>param_name</i>	Parameter name
in	<i>data_type</i>	Parameter type
out	<i>value</i>	Parameter value

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

- [fortran/new/cwp\\_f.f90](#)

## 3.61 cwp::cwp\_param\_is Interface Reference

### Public Member Functions

- integer function [cwp\\_param\\_is\\_](#) (*code\_name*, *param\_name*, *data\_type*)  
*Is this code\_name a parameter ?*

### 3.61.1 Member Function/Subroutine Documentation

#### 3.61.1.1 cwp\_param\_is\_()

```
integer function cwp::cwp_param_is::cwp_param_is_ (
    character(kind = c_char, len = *) code_name,
    character(kind = c_char, len = *) param_name,
    integer, intent(in) data_type )
```

Is this *code\_name* a parameter ?

**Parameters**

in	<i>code_name</i>	Local or distant code name
in	<i>param_name</i>	Parameter name
in	<i>data_type</i>	Parameter type,

return 1 : true / 0 : false

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

- [fortran/new/cwp\\_f.f90](#)

## 3.62 cwp::cwp\_param\_list\_get Interface Reference

### Public Member Functions

- subroutine [cwp\\_param\\_list\\_get\\_](#) (code\_name, data\_type, n\_param, param\_names)  
*Return the list of parameters for the code code\_name.*

### 3.62.1 Member Function/Subroutine Documentation

#### 3.62.1.1 cwp\_param\_list\_get\_()

```
subroutine cwp::cwp_param_list_get::cwp_param_list_get_ (
    character(kind = c_char, len = *) code_name,
    integer data_type,
    integer(c_int) n_param,
    character(256), dimension(:), allocatable param_names )
```

Return the list of parameters for the code code\_name.

#### Parameters

in	<i>code_name</i>	Local or distant code name
in	<i>data_type</i>	Parameter type
in	<i>n_param</i>	Number of parameters
in	<i>param_names</i>	Parameter names

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

- [fortran/new/cwp\\_f.f90](#)

## 3.63 cwp::cwp\_param\_lock Interface Reference

### Public Member Functions

- subroutine [cwp\\_param\\_lock\\_](#) (code\_name)  
*Lock access to local parameters from a distant code.*

### 3.63.1 Member Function/Subroutine Documentation

#### 3.63.1.1 cwp\_param\_lock\_()

```
subroutine cwp::cwp_param_lock::cwp_param_lock_ (
    character(kind = c_char, len = *) code_name )
```

Lock access to local parameters from a distant code.

## Parameters

in	<i>code_name</i>	Code to lock
----	------------------	--------------

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

- [fortran/new/cwp\\_f.f90](#)

## 3.64 cwp::cwp\_param\_n\_get Interface Reference

### Public Member Functions

- integer function [cwp\\_param\\_n\\_get\\_](#) (*code\_name*, *data\_type*)  
*Return the number of parameters for the code code\_name.*

### 3.64.1 Member Function/Subroutine Documentation

#### 3.64.1.1 cwp\_param\_n\_get\_()

```
integer function cwp::cwp_param_n_get::cwp_param_n_get_ (
    character(kind = c_char, len = *) code_name,
    integer, intent(in) data_type )
```

Return the number of parameters for the code *code\_name*.

## Parameters

in	<i>code_name</i>	Local or distant code name
in	<i>data_type</i>	Parameter type,

return Number of parameters

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

- [fortran/new/cwp\\_f.f90](#)

## 3.65 cwp::cwp\_param\_reduce Interface Reference

### Public Member Functions

- subroutine [cwp\\_param\\_reduce\\_int](#) (*op*, *param\_name*, *res*, *n\_codes*, *code\_names*)  
*Return the result of a reduce operation about a parameter.*
- subroutine [cwp\\_param\\_reduce\\_double](#) (*op*, *param\_name*, *res*, *n\_codes*, *code\_names*)
- subroutine [cwp\\_param\\_reduce\\_char](#) (*op*, *param\_name*, *res*, *n\_codes*, *code\_names*)



### 3.65.1 Member Function/Subroutine Documentation

#### 3.65.1.1 cwp\_param\_reduce\_int()

```
subroutine cwp::cwp_param_reduce::cwp_param_reduce_int (
    integer, intent(in) op,
    character(kind = c_char, len = *) param_name,
    integer(c_int), intent(out) res,
    integer(c_int) n_codes,
    character(kind = c_char, len = *), dimension(n_codes), target code_names )
```

Return the result of a reduce operation about a parameter.

##### Parameters

in	<i>op</i>	Operation
in	<i>param_name</i>	Parameter name
in	<i>data_type</i>	Parameter type
out	<i>res</i>	Result
in	<i>n_codes</i>	Number of codes
in	<i>code_names</i>	Codes name

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

- [fortran/new/cwp\\_f.f90](#)

## 3.66 cwp::cwp\_param\_set Interface Reference

### Public Member Functions

- subroutine [cwp\\_param\\_set\\_int\\_](#) (*local\_code\_name*, *param\_name*, *value*)  
*Set a parameter.*
- subroutine [cwp\\_param\\_set\\_double\\_](#) (*local\_code\_name*, *param\_name*, *value*)
- subroutine [cwp\\_param\\_set\\_char\\_](#) (*local\_code\_name*, *param\_name*, *value*)

### 3.66.1 Member Function/Subroutine Documentation

#### 3.66.1.1 cwp\_param\_set\_int\_()

```
subroutine cwp::cwp_param_set::cwp_param_set_int_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) param_name,
    integer(kind = c_int), intent(in) value )
```

Set a parameter.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>param_name</i>	Parameter name
in	<i>data_type</i>	Parameter type
in	<i>value</i>	Value

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

- [fortran/new/cwp\\_f.f90](#)

## 3.67 cwp::cwp\_param\_unlock Interface Reference

### Public Member Functions

- subroutine [cwp\\_param\\_unlock\\_](#) (code\_name)  
*Unlock access to local parameters from a distant code.*

#### 3.67.1 Member Function/Subroutine Documentation

##### 3.67.1.1 cwp\_param\_unlock\_()

```
subroutine cwp::cwp_param_unlock::cwp_param_unlock_ (
    character(kind = c_char, len = *) code_name )
```

Unlock access to local parameters from a distant code.

## Parameters

in	<i>code_name</i>	Code to unlock
----	------------------	----------------

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

- [fortran/new/cwp\\_f.f90](#)

## 3.68 cwp::cwp\_part\_data\_create Interface Reference

### Public Member Functions

- subroutine [cwp\\_part\\_data\\_create\\_](#) (local\_code\_name, cpl\_id, part\_data\_id, exch\_type, gnum\_elt, n\_elt, n↔\_part)  
*Create partitioned data exchange object.*

### 3.68.1 Member Function/Subroutine Documentation

#### 3.68.1.1 cwp\_part\_data\_create\_()

```
subroutine cwp::cwp_part_data_create::cwp_part_data_create_ (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id,
    character(kind=c_char, len=*) part_data_id,
    integer(c_int), intent(in)  exch_type,
    type(pdm_pointer_array_t), target gnum_elt,
    integer(c_int), dimension(:), pointer n_elt,
    integer, intent(in) n_part )
```

Create partitioned data exchange object.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>part_data_id</i>	PartData identifier
in	<i>exch_type</i>	Exchange type
in	<i>gnum_elt</i>	Global ids
in	<i>n_elt</i>	Number of elements in partitions (size = <i>n_part</i> )
in	<i>n_part</i>	Number of partitions

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

- [fortran/new/cwp\\_f.f90](#)

## 3.69 cwp::cwp\_part\_data\_del Interface Reference

### Public Member Functions

- subroutine [cwp\\_part\\_data\\_del\\_](#) (*local\_code\_name*, *cpl\_id*, *part\_data\_id*)  
*Delete partitioned data exchange object.*

### 3.69.1 Member Function/Subroutine Documentation

#### 3.69.1.1 cwp\_part\_data\_del\_()

```
subroutine cwp::cwp_part_data_del::cwp_part_data_del_ (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id,
    character(kind=c_char, len=*) part_data_id )
```

Delete partitioned data exchange object.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>part_data_id</i>	PartData identifier

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

- fortran/new/[cwp\\_f.f90](#)

## 3.70 cwp::cwp\_part\_data\_irecv Interface Reference

### Public Member Functions

- subroutine [cwp\\_part\\_data\\_irecv\\_](#) (*local\_code\_name*, *cpl\_id*, *part\_data\_id*, *exch\_id*, *n\_components*, *recv\_↔data*)

*Receive a data array.*

### 3.70.1 Member Function/Subroutine Documentation

#### 3.70.1.1 cwp\_part\_data\_irecv\_()

```
subroutine cwp::cwp_part_data_irecv::cwp_part_data_irecv_ (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id,
    character(kind=c_char, len=*) part_data_id,
    integer(c_int), intent(in) exch_id,
    integer(c_int), intent(in) n_components,
    type(pdm_pointer_array_t), target recv_data )
```

Receive a data array.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>part_data_id</i>	PartData identifier
in	<i>exch_id</i>	Exchange identifier
in	<i>n_components</i>	Number of components
in, out	<i>recv_data</i>	Pointer to data array to receive

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

- fortran/new/[cwp\\_f.f90](#)

## 3.71 cwp::cwp\_part\_data\_issend Interface Reference

### Public Member Functions

- subroutine [cwp\\_part\\_data\\_issend](#) (local\_code\_name, cpl\_id, part\_data\_id, exch\_id, n\_components, send\_data)

*Send a data array.*

### 3.71.1 Member Function/Subroutine Documentation

#### 3.71.1.1 cwp\_part\_data\_issend\_()

```
subroutine cwp::cwp_part_data_issend::cwp_part_data_issend_ (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id,
    character(kind=c_char, len=*) part_data_id,
    integer(c_int), intent(in) exch_id,
    integer(c_int), intent(in) n_components,
    type(pdm_pointer_array_t), target send_data )
```

Send a data array.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>part_data_id</i>	PartData identifier
in	<i>exch_id</i>	Exchange identifier
in	<i>n_components</i>	Number of components
in	<i>send_data</i>	Pointer to data array to send

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

- [fortran/new/cwp\\_f.f90](#)

## 3.72 cwp::cwp\_part\_data\_wait\_irecv Interface Reference

### Public Member Functions

- subroutine [cwp\\_part\\_data\\_wait\\_irecv](#) (local\_code\_name, cpl\_id, part\_data\_id, exch\_id)

*Wait of receive a data array.*

### 3.72.1 Member Function/Subroutine Documentation

### 3.72.1.1 cwp\_part\_data\_wait\_irecv\_()

```
subroutine cwp::cwp_part_data_wait_irecv::cwp_part_data_wait_irecv_ (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id,
    character(kind=c_char, len=*) part_data_id,
    integer(c_int), intent(in) exch_id )
```

Wait of receive a data array.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>part_data_id</i>	PartData identifier
in	<i>exch_id</i>	Exchange identifier

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

- [fortran/new/cwp\\_f.f90](#)

## 3.73 cwp::cwp\_part\_data\_wait\_issend Interface Reference

### Public Member Functions

- subroutine [cwp\\_part\\_data\\_wait\\_issend\\_](#) (local\_code\_name, cpl\_id, part\_data\_id, exch\_id)  
*Wait of send a data array.*

### 3.73.1 Member Function/Subroutine Documentation

#### 3.73.1.1 cwp\_part\_data\_wait\_issend\_()

```
subroutine cwp::cwp_part_data_wait_issend::cwp_part_data_wait_issend_ (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id,
    character(kind=c_char, len=*) part_data_id,
    integer(c_int), intent(in) exch_id )
```

Wait of send a data array.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>part_data_id</i>	PartData identifier
in	<i>exch_id</i>	Exchange identifier

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

- [fortran/new/cwp\\_f.f90](#)

## 3.74 cwp::CWP\_Properties\_dump Interface Reference

Dump code properties.

### Public Member Functions

- subroutine [cwp\\_properties\\_dump](#) ()

### 3.74.1 Detailed Description

Dump code properties.

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

- [fortran/new/cwp\\_f.f90](#)

## 3.75 cwp::cwp\_spatial\_interp\_property\_set Interface Reference

### Public Member Functions

- subroutine [cwp\\_spatial\\_interp\\_property\\_set\\_](#) (local\_code\_name, cpl\_id, property\_name, property\_type, property\_value)

*Set a property of the spatial interpolation algorithm.*

### 3.75.1 Member Function/Subroutine Documentation

#### 3.75.1.1 cwp\_spatial\_interp\_property\_set\_()

```
subroutine cwp::cwp_spatial_interp_property_set::cwp_spatial_interp_property_set_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) property_name,
    integer (kind = c_int) property_type,
    character(kind = c_char, len = *) property_value )
```

Set a property of the spatial interpolation algorithm.

**Parameters**

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>property_name</i>	Name of the property
in	<i>property_type</i>	Type of the property
in	<i>property_value</i>	Value of the property

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

- [fortran/new/cwp\\_f.f90](#)

## 3.76 cwp::cwp\_spatial\_interp\_weights\_compute Interface Reference

### Public Member Functions

- subroutine [cwp\\_spatial\\_interp\\_weights\\_compute\\_](#) (local\_code\_name, cpl\_id)  
*Compute spatial interpolation weights.*

### 3.76.1 Member Function/Subroutine Documentation

#### 3.76.1.1 cwp\_spatial\_interp\_weights\_compute\_()

```
subroutine cwp::cwp_spatial_interp_weights_compute::cwp_spatial_interp_weights_compute_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id )
```

Compute spatial interpolation weights.

**Parameters**

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier

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

- [fortran/new/cwp\\_f.f90](#)

## 3.77 cwp::cwp\_state\_get Interface Reference

### Public Member Functions

- integer(c\_int) function [cwp\\_state\\_get\\_](#) (local\_code\_name)  
*Return code state.*



### 3.77.1 Member Function/Subroutine Documentation

#### 3.77.1.1 cwp\_state\_get\_()

```
integer(c_int) function cwp::cwp_state_get::cwp_state_get_ (
    character(kind = c_char, len = *) local_code_name )
```

Return code state.

##### Parameters

in	<i>code_name</i>	Code name
----	------------------	-----------

##### Returns

Code state

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

- [fortran/new/cwp\\_f.f90](#)

## 3.78 cwp::cwp\_state\_update Interface Reference

### Public Member Functions

- subroutine [cwp\\_state\\_update\\_](#) (local\_code\_name, state)  
*Update code state.*

### 3.78.1 Member Function/Subroutine Documentation

#### 3.78.1.1 cwp\_state\_update\_()

```
subroutine cwp::cwp_state_update::cwp_state_update_ (
    character(kind = c_char, len = *) local_code_name,
    integer(kind = c_int), intent(in) state )
```

Update code state.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>state</i>	State

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

- [fortran/new/cwp\\_f.f90](#)

## 3.79 cwp::cwp\_time\_step\_beg Interface Reference

### Public Member Functions

- subroutine [cwp\\_time\\_step\\_beg\\_](#) (local\_code\_name, current\_time)  
*Begin code time step.*

### 3.79.1 Member Function/Subroutine Documentation

#### 3.79.1.1 cwp\_time\_step\_beg\_()

```
subroutine cwp::cwp_time_step_beg::cwp_time_step_beg_ (
    character(kind = c_char, len = *) local_code_name,
    double precision, intent(in) current_time )
```

Begin code time step.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>current_time</i>	Current time

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

- [fortran/new/cwp\\_f.f90](#)

## 3.80 cwp::cwp\_time\_step\_end Interface Reference

### Public Member Functions

- subroutine [cwp\\_time\\_step\\_end\\_](#) (local\_code\_name)  
*End code time step.*

### 3.80.1 Member Function/Subroutine Documentation

### 3.80.1.1 cwp\_time\_step\_end\_()

```
subroutine cwp::cwp_time_step_end::cwp_time_step_end_ (
    character(kind = c_char, len = *) local_code_name )
```

End code time step.

#### Parameters

in	<i>local_code_name</i>	Local code name
----	------------------------	-----------------

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

- [fortran/new/cwp\\_f.f90](#)

## 3.81 cwp::cwp\_uncomputed\_tgts\_get Interface Reference

### Public Member Functions

- integer(c\_int) function, dimension(:), pointer [cwp\\_uncomputed\\_tgts\\_get\\_](#) (local\_code\_name, cpl\_id, field\_id, i\_part)

*Return uncomputed targets.*

### 3.81.1 Member Function/Subroutine Documentation

#### 3.81.1.1 cwp\_uncomputed\_tgts\_get\_()

```
integer(c_int) function, dimension(:), pointer cwp::cwp_uncomputed_tgts_get::cwp_uncomputed_↵
tgts_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part )
```

Return uncomputed targets.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Current partition

**Returns**

Uncomputed targets

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

- [fortran/new/cwp\\_f.f90](#)

## 3.82 cwp::cwp\_user\_structure\_get Interface Reference

### Public Member Functions

- `type(c_ptr) function cwp_user_structure_get_ (local_code_name)`  
*Return the user structure associated.*

### 3.82.1 Member Function/Subroutine Documentation

#### 3.82.1.1 cwp\_user\_structure\_get\_()

```
type(c_ptr) function cwp::cwp_user_structure_get::cwp_user_structure_get_ (
    character(kind = c_char, len = *) local_code_name )
```

Return the user structure associated.

This structure can be called into a callback

#### Parameters

in	<i>local_code_name</i>	Local code name
----	------------------------	-----------------

**Returns**

User structure

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

- [fortran/new/cwp\\_f.f90](#)

## 3.83 cwp::cwp\_user\_structure\_set Interface Reference

### Public Member Functions

- subroutine **cwp\_user\_structure\_set\_** (local\_code\_name, user\_structure)

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

- [fortran/new/cwp\\_f.f90](#)

## 3.84 cwp::cwp\_user\_tgt\_pts\_set Interface Reference

### Public Member Functions

- subroutine [cwp\\_user\\_tgt\\_pts\\_set](#) (local\_code\_name, cpl\_id, i\_part, n\_pts, coord, global\_num)  
*Setting user target points.*

### 3.84.1 Member Function/Subroutine Documentation

#### 3.84.1.1 cwp\_user\_tgt\_pts\_set\_()

```
subroutine cwp::cwp_user_tgt_pts_set::cwp_user_tgt_pts_set_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    integer(kind = c_int) i_part,
    integer(kind = c_int) n_pts,
    double precision, dimension(:, :), pointer coord,
    integer(kind = c_long), dimension(:), pointer global_num )
```

Setting user target points.

This function must be called if the degrees of freedom locations are CWP\_DOF\_LOCATION\_USER

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Current partition
in	<i>n_pts</i>	Number of points
in	<i>coord</i>	Coordinates (size = 3 * n_pts)
in	<i>g_num</i>	global number or NUL (size = n_pts)

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

- [fortran/new/cwp\\_f.f90](#)

## 3.85 cwp::cwp\_visu\_set Interface Reference

### Public Member Functions

- subroutine [cwp\\_visu\\_set](#) (local\_code\_name, cpl\_id, freq, format, format\_option)  
*Enable visualization output.*

### 3.85.1 Member Function/Subroutine Documentation

#### 3.85.1.1 cwp\_visu\_set\_()

```
subroutine cwp::cwp_visu_set::cwp_visu_set_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    integer(c_int) freq,
    integer(c_int) format,
    character(kind = c_char, len = *) format_option )
```

Enable visualization output.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>freq</i>	Output frequency
in	<i>format</i>	Output format to visualize exchanged fieldsDouble on the coupled mesh. Choice between : <ul style="list-style-type: none"> <li>• "EnSight Gold"</li> </ul>
in	<i>format_option</i>	Output options "opt1, opt2, ..." <ul style="list-style-type: none"> <li>• text : output text files</li> <li>• binary : output binary files (default)</li> </ul>

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

- [fortran/new/cwp\\_f.f90](#)

## Chapter 4

# File Documentation

### 4.1 fortran/new/cwp\_f.f90 File Reference

#### Data Types

- interface [cwp::cwp\\_param\\_set](#)
- interface [cwp::cwp\\_param\\_add](#)
- interface [cwp::cwp\\_init](#)
- interface [cwp::cwp\\_c\\_to\\_f\\_string](#)
- interface [cwp::cwp\\_state\\_update](#)
- interface [cwp::cwp\\_time\\_step\\_beg](#)
- interface [cwp::cwp\\_time\\_step\\_end](#)
- interface [cwp::cwp\\_user\\_structure\\_set](#)
- interface [cwp::cwp\\_user\\_structure\\_get](#)
- interface [cwp::cwp\\_output\\_file\\_set](#)
- interface [cwp::cwp\\_state\\_get](#)
- interface [cwp::cwp\\_cpl\\_create](#)
- interface [cwp::cwp\\_cpl\\_barrier](#)
- interface [cwp::cwp\\_cpl\\_del](#)
- interface [cwp::cwp\\_computed\\_tgts\\_bcast\\_enable](#)
- interface [cwp::cwp\\_involved\\_srcs\\_bcast\\_enable](#)
- interface [cwp::cwp\\_n\\_uncomputed\\_tgts\\_get](#)
- interface [cwp::cwp\\_uncomputed\\_tgts\\_get](#)
- interface [cwp::cwp\\_n\\_computed\\_tgts\\_get](#)
- interface [cwp::cwp\\_computed\\_tgts\\_get](#)
- interface [cwp::cwp\\_n\\_involved\\_srcs\\_get](#)
- interface [cwp::cwp\\_involved\\_srcs\\_get](#)
- interface [cwp::cwp\\_spatial\\_interp\\_weights\\_compute](#)
- interface [cwp::cwp\\_spatial\\_interp\\_property\\_set](#)
- interface [cwp::cwp\\_visu\\_set](#)
- interface [cwp::cwp\\_user\\_tgt\\_pts\\_set](#)
- interface [cwp::cwp\\_mesh\\_interf\\_finalize](#)
- interface [cwp::cwp\\_mesh\\_interf\\_vtx\\_set](#)
- interface [cwp::cwp\\_mesh\\_interf\\_block\\_add](#)
- interface [cwp::cwp\\_mesh\\_interf\\_block\\_std\\_set](#)
- interface [cwp::cwp\\_mesh\\_interf\\_block\\_std\\_get](#)
- interface [cwp::cwp\\_mesh\\_interf\\_f\\_poly\\_block\\_set](#)
- interface [cwp::cwp\\_mesh\\_interf\\_f\\_poly\\_block\\_get](#)

- interface `cwp::cwp_mesh_interf_c_poly_block_set`
- interface `cwp::cwp_mesh_interf_c_poly_block_get`
- interface `cwp::cwp_mesh_interf_del`
- interface `cwp::cwp_mesh_interf_from_cellface_set`
- interface `cwp::cwp_mesh_interf_from_faceedge_set`
- interface `cwp::cwp_field_create`
- interface `cwp::cwp_field_data_set`
- interface `cwp::cwp_field_dof_location_get`
- interface `cwp::cwp_field_storage_get`
- interface `cwp::cwp_field_n_dof_get`
- interface `cwp::cwp_field_del`
- interface `cwp::cwp_field_issend`
- interface `cwp::cwp_field_irecv`
- interface `cwp::cwp_field_wait_issend`
- interface `cwp::cwp_field_wait_irecv`
- interface `cwp::cwp_field_interp_function_unset`
- interface `cwp::cwp_field_interp_function_set`
- interface `cwp::cwp_field_n_components_get`
- interface `cwp::cwp_field_src_data_properties_get`
- interface `cwp::cwp_field_tgt_data_properties_get`
- interface `cwp::cwp_field_location_weights_get`
- interface `cwp::cwp_field_location_point_data_get`
- interface `cwp::cwp_field_location_internal_cell_vtx_get`
- interface `cwp::cwp_field_intersection_volumes_get`
- interface `cwp::cwp_field_intersection_tgt_elt_volumes_get`
- interface `cwp::cwp_field_nearest_neighbors_distances_get`
- interface `cwp::cwp_field_nearest_neighbors_coord_get`
- interface `cwp::cwp_param_del`
- interface `cwp::cwp_param_n_get`
- interface `cwp::cwp_param_is`
- interface `cwp::cwp_param_get`
- interface `cwp::cwp_param_reduce`
- interface `cwp::cwp_param_lock`
- interface `cwp::cwp_param_unlock`
- interface `cwp::cwp_codes_list_get`
- interface `cwp::cwp_loc_codes_list_get`
- interface `cwp::cwp_param_list_get`
- interface `cwp::cwp_global_data_issend`
- interface `cwp::cwp_global_data_irecv`
- interface `cwp::cwp_global_data_wait_issend`
- interface `cwp::cwp_global_data_wait_irecv`
- interface `cwp::cwp_part_data_create`
- interface `cwp::cwp_part_data_del`
- interface `cwp::cwp_part_data_issend`
- interface `cwp::cwp_part_data_irecv`
- interface `cwp::cwp_part_data_wait_issend`
- interface `cwp::cwp_part_data_wait_irecv`
- interface `cwp::cwp_cpl_spatial_interp_algo_get`
- interface `cwp::CWP_Finalize`  
*Finalize CWIPI.*
- interface `cwp::CWP_Codes_nb_get`  
*Return the number of codes known by CWIPI.*
- interface `cwp::CWP_Loc_codes_nb_get`  
*Return the number of local codes known by CWIPI.*
- interface `cwp::CWP_Properties_dump`  
*Dump code properties.*



## Enumerations

- enum { **cwp\_double** , **cwp\_int** , **cwp\_char** }
- enum { **cwp\_visu\_format\_ensight** }
- enum { **cwp\_comm\_par\_with\_part** , **cwp\_comm\_par\_without\_part** }
- enum { **cwp\_time\_exch\_user\_controlled** }
- enum { **cwp\_dof\_location\_undef** , **cwp\_dof\_location\_cell\_center** , **cwp\_dof\_location\_node** , **cwp\_dof\_location\_user** }
- enum { **cwp\_field\_exch\_send** , **cwp\_field\_exch\_rcv** , **cwp\_field\_exch\_sendrcv** }
- enum { **cwp\_field\_map\_source** , **cwp\_field\_map\_target** }
- enum { **cwp\_field\_storage\_interlaced** , **cwp\_field\_storage\_interleaved** }
- enum { **cwp\_block\_node** , **cwp\_block\_edge2** , **cwp\_block\_face\_tria3** , **cwp\_block\_face\_quad4** , **cwp\_block\_face\_poly** , **cwp\_block\_cell\_tetra4** , **cwp\_block\_cell\_hexa8** , **cwp\_block\_cell\_prism6** , **cwp\_block\_cell\_pyram5** , **cwp\_block\_cell\_poly** }
- enum { **cwp\_spatial\_interp\_from\_nearest\_sources\_least\_squares** , **cwp\_spatial\_interp\_from\_nearest\_targets\_least\_squares** , **cwp\_spatial\_interp\_from\_intersection** , **cwp\_spatial\_interp\_from\_location\_mesh\_location\_locate\_all\_tgt** , **cwp\_spatial\_interp\_from\_location\_mesh\_location\_octree** , **cwp\_spatial\_interp\_from\_location\_mesh\_location\_boxtree** , **cwp\_spatial\_interp\_from\_identity** }
- enum { **cwp\_interface\_point** , **cwp\_interface\_linear** , **cwp\_interface\_surface** , **cwp\_interface\_volume** }
- enum { **cwp\_dynamic\_mesh\_static** , **cwp\_dynamic\_mesh\_deformable** , **cwp\_dynamic\_mesh\_variable** }
- enum { **cwp\_status\_off** , **cwp\_status\_on** }
- enum { **cwp\_err\_no\_error** , **cwp\_err\_default** }
- enum { **cwp\_state\_in\_progress** , **cwp\_state\_end** , **cwp\_state\_output\_error** }
- enum { **cwp\_op\_min** , **cwp\_op\_max** , **cwp\_op\_sum** }
- enum { **cwp\_partdata\_send** , **cwp\_partdata\_rcv** }

## Functions/Subroutines

- subroutine, private **cwp::c\_f\_char\_array** (c\_char\_array, c\_size\_array, n\_chars, f\_char\_array, free\_all)  
*convert an array of char \* in c to an array*
- subroutine, private **cwp::cwp\_init** (fcomm, n\_code, code\_names, is\_active\_rank, intra\_comms)  
*Initialize CWIPI.*
- character(len=:) function, pointer, private **cwp::cwp\_c\_to\_f\_string** (c\_str)  
*Create a Fortran string from a C string.*
- subroutine, private **cwp::cwp\_state\_update** (local\_code\_name, state)  
*Update code state.*
- subroutine, private **cwp::cwp\_time\_step\_beg** (local\_code\_name, current\_time)  
*Begin code time step.*
- subroutine, private **cwp::cwp\_time\_step\_end** (local\_code\_name)  
*End code time step.*
- subroutine, private **cwp::cwp\_user\_structure\_set** (local\_code\_name, user\_structure)
- type(c\_ptr) function, private **cwp::cwp\_user\_structure\_get** (local\_code\_name)  
*Return the user structure associated.*
- subroutine **cwp::cwp\_output\_fortran\_unit\_set** (outputUnit)  
*Writing output to Fortran file (shared by fortran and C code).*
- subroutine, private **cwp::cwp\_output\_file\_set** (f\_output\_file\_name)  
*Define output file (in which only C code writes).*
- integer(c\_int) function, private **cwp::cwp\_state\_get** (local\_code\_name)  
*Return code state.*

- character(256) function, dimension(:), allocatable, private [cwp::cwp\\_codes\\_list\\_get\\_\(\)](#)  
*Return list of codes known by CWIPI.*
- character(256) function, dimension(:), allocatable, private [cwp::cwp\\_loc\\_codes\\_list\\_get\\_\(\)](#)  
*Return list of local codes known by CWIPI.*
- subroutine, private [cwp::cwp\\_cpl\\_create\\_\(\)](#) (local\_code\_name, cpl\_id, coupled\_code\_name, entities\_dim, comm\_type, spatial\_interp, n\_part, displacement, freq)  
*Create a coupling object and define its properties.*
- subroutine, private [cwp::cwp\\_cpl\\_barrier\\_\(\)](#) (local\_code\_name, cpl\_id)  
*MPI Barrier on the coupling communicator.*
- subroutine, private [cwp::cwp\\_cpl\\_del\\_\(\)](#) (local\_code\_name, cpl\_id)  
*Delete a coupling object.*
- subroutine, private [cwp::cwp\\_computed\\_tgts\\_bcast\\_enable\\_\(\)](#) (local\_code\_name, cpl\_id, field\_id)  
*Enable broadcast of the computed targets ids (in CWP\_COMM\_PAR\_WITHOUT\_PART mode).*
- subroutine, private [cwp::cwp\\_involved\\_srcs\\_bcast\\_enable\\_\(\)](#) (local\_code\_name, cpl\_id, field\_id)  
*Enable broadcast of the involved sources ids (in CWP\_COMM\_PAR\_WITHOUT\_PART mode).*
- integer(c\_int) function, private [cwp::cwp\\_n\\_uncomputed\\_tgts\\_get\\_\(\)](#) (local\_code\_name, cpl\_id, field\_id, i\_part)  
*Return the number of uncomputed targets.*
- integer(c\_int) function, dimension(:), pointer, private [cwp::cwp\\_uncomputed\\_tgts\\_get\\_\(\)](#) (local\_code\_name, cpl\_id, field\_id, i\_part)  
*Return uncomputed targets.*
- integer(c\_int) function, private [cwp::cwp\\_n\\_computed\\_tgts\\_get\\_\(\)](#) (local\_code\_name, cpl\_id, field\_id, i\_part)  
*Return the number of computed targets.*
- integer(c\_int) function, dimension(:), pointer, private [cwp::cwp\\_computed\\_tgts\\_get\\_\(\)](#) (local\_code\_name, cpl\_id, field\_id, i\_part)  
*Return computed targets.*
- integer(c\_int) function, private [cwp::cwp\\_n\\_involved\\_srcs\\_get\\_\(\)](#) (local\_code\_name, cpl\_id, field\_id, i\_part)  
*Return the number of involved sources.*
- integer(c\_int) function, dimension(:), pointer, private [cwp::cwp\\_involved\\_srcs\\_get\\_\(\)](#) (local\_code\_name, cpl\_id, field\_id, i\_part)  
*Return involved sources.*
- subroutine, private [cwp::cwp\\_spatial\\_interp\\_weights\\_compute\\_\(\)](#) (local\_code\_name, cpl\_id)  
*Compute spatial interpolation weights.*
- subroutine, private [cwp::cwp\\_spatial\\_interp\\_property\\_set\\_\(\)](#) (local\_code\_name, cpl\_id, property\_name, property\_type, property\_value)  
*Set a property of the spatial interpolation algorithm.*
- subroutine, private [cwp::cwp\\_visu\\_set\\_\(\)](#) (local\_code\_name, cpl\_id, freq, format, format\_option)  
*Enable visualization output.*
- subroutine, private [cwp::cwp\\_user\\_tgt\\_pts\\_set\\_\(\)](#) (local\_code\_name, cpl\_id, i\_part, n\_pts, coord, global\_num)  
*Setting user target points.*
- subroutine, private [cwp::cwp\\_mesh\\_intf\\_finalize\\_\(\)](#) (local\_code\_name, cpl\_id)  
*Finalize interface mesh.*
- subroutine, private [cwp::cwp\\_mesh\\_intf\\_vtx\\_set\\_\(\)](#) (local\_code\_name, cpl\_id, i\_part, n\_pts, coord, global\_num)  
*Set vertices.*
- integer(c\_int) function, private [cwp::cwp\\_mesh\\_intf\\_block\\_add\\_\(\)](#) (local\_code\_name, cpl\_id, block\_type)  
*Add a connectivity block to the interface mesh.*
- subroutine, private [cwp::cwp\\_mesh\\_intf\\_block\\_std\\_set\\_\(\)](#) (local\_code\_name, cpl\_id, i\_part, block\_id, n\_elts, connec, global\_num)  
*Set a standard block to the interface mesh.*
- subroutine, private [cwp::cwp\\_mesh\\_intf\\_block\\_std\\_get\\_\(\)](#) (local\_code\_name, cpl\_id, i\_part, block\_id, n\_elts, connec, global\_num)  
*Get the properties of a standard block of the interface mesh.*

- subroutine, private `cwp::cwp_mesh_interf_f_poly_block_set_` (local\_code\_name, cpl\_id, i\_part, block\_id, n↔\_elts, connec\_idx, connec, global\_num)  
*Set the connectivity of a polygon block in a interface mesh partition.*
- subroutine, private `cwp::cwp_mesh_interf_f_poly_block_get_` (local\_code\_name, cpl\_id, i\_part, block\_id, n↔\_elts, connec\_idx, connec, global\_num)  
*Get the properties of a polygon block of the interface mesh partition.*
- subroutine, private `cwp::cwp_mesh_interf_c_poly_block_set_` (local\_code\_name, cpl\_id, i\_part, block\_id, n↔\_elts, n\_faces, connec\_faces\_idx, connec\_faces, connec\_cells\_idx, connec\_cells, global\_num)  
*Adding a polyhedron connectivity block to the interface mesh.*
- subroutine, private `cwp::cwp_mesh_interf_c_poly_block_get_` (local\_code\_name, cpl\_id, i\_part, block\_id, n↔\_elts, n\_faces, connec\_faces\_idx, connec\_faces, connec\_cells\_idx, connec\_cells, global\_num)  
*Get the properties of a polyhedron block of the interface mesh partition..*
- subroutine, private `cwp::cwp_mesh_interf_del_` (local\_code\_name, cpl\_id)  
*Delete interface mesh.*
- subroutine, private `cwp::cwp_mesh_interf_from_cellface_set_` (local\_code\_name, cpl\_id, i\_part, n\_cells, cell\_face\_idx, cell\_face, n\_faces, face\_vtx\_idx, face\_vtx, global\_num)  
*Define the interface mesh from a cell to face connectivity.*
- subroutine, private `cwp::cwp_mesh_interf_from_faceedge_set_` (local\_code\_name, cpl\_id, i\_part, n\_faces, face\_edge\_idx, face\_edge, n\_edges, edge\_vtx, global\_num)  
*Define the surface interface mesh from a face to edge connectivity.*
- subroutine, private `cwp::cwp_field_create_` (local\_code\_name, cpl\_id, field\_id, data\_type, storage, n↔component, target\_location, exch\_type, visu\_status)  
*Create a new field.*
- subroutine, private `cwp::cwp_field_data_set_` (local\_code\_name, cpl\_id, field\_id, i\_part, map\_type, data)  
*Set field data.*
- integer(c\_int) function, private `cwp::cwp_field_dof_location_get_` (local\_code\_name, cpl\_id, field\_id)  
*Get target degrees of freedom location.*
- integer(c\_int) function, private `cwp::cwp_field_storage_get_` (local\_code\_name, cpl\_id, field\_id)  
*Get field storage type.*
- integer(c\_int) function, private `cwp::cwp_field_n_dof_get_` (local\_code\_name, cpl\_id, field\_id, i\_part)  
*Get field number of degrees of freedom.*
- subroutine, private `cwp::cwp_field_del_` (local\_code\_name, cpl\_id, field\_id)  
*Delete a field.*
- subroutine, private `cwp::cwp_field_issend_` (local\_code\_name, cpl\_id, field\_id)  
*Send a spatially interpolated field to the coupled code with non-blocking communications.*
- subroutine, private `cwp::cwp_field_irecv_` (local\_code\_name, cpl\_id, tgt\_field\_id)  
*Receive a spatially interpolated field from the coupled code with non-blocking communications.*
- subroutine, private `cwp::cwp_field_wait_issend_` (local\_code\_name, cpl\_id, field\_id)  
*Wait the end of an exchange related to request from CWP\_Field\_issend.*
- subroutine, private `cwp::cwp_field_wait_irecv_` (local\_code\_name, cpl\_id, tgt\_field\_id)  
*Wait the end of an exchange related to request from CWP\_Field\_irecv.*
- subroutine, private `cwp::cwp_field_interp_function_unset_` (local\_code\_name, cpl\_id, field\_id)  
*Unsetting of a user interpolation.*
- subroutine, private `cwp::cwp_field_interp_function_set_` (local\_code\_name, cpl\_id, field\_id, user↔interpolation\_fct)  
*Setting of a user interpolation from location.*
- integer function, private `cwp::cwp_field_n_components_get_` (local\_code\_name, cpl\_id, field\_id)  
*Get spatial interpolation number of algorithms.*
- subroutine, private `cwp::cwp_field_src_data_properties_get_` (local\_code\_name, cpl\_id, field\_id, i\_part, n↔\_elt\_src, src\_to\_tgt\_idx)  
*Get spatial interpolation source data.*

- subroutine, private `cwp::cwp_field_tgt_data_properties_get_` (local\_code\_name, cpl\_id, field\_id, i\_part, n\_↵  
elt\_tgt, n\_referenced\_tgt, referenced\_tgt, tgt\_come\_from\_src\_idx)  
*Get spatial interpolation target data.*
- subroutine, private `cwp::cwp_field_location_weights_get_` (local\_code\_name, cpl\_id, field\_id, i\_part, weights)  
*Get spatial interpolation weights (location algorithm).*
- subroutine, private `cwp::cwp_field_location_point_data_get_` (local\_code\_name, cpl\_id, field\_id, i\_part,  
points\_coords, points\_uvw, points\_dist2, points\_projected\_coords)  
*Get spatial interpolation point data (location algorithm).*
- subroutine, private `cwp::cwp_field_location_internal_cell_vtx_get_` (local\_code\_name, cpl\_id, field\_id, i\_part,  
cell\_vtx\_idx, cell\_vtx)  
*Get spatial interpolation internal cell-> vertex connectivity (location algorithm).*
- subroutine, private `cwp::cwp_field_intersection_volumes_get_` (local\_code\_name, cpl\_id, field\_id, i\_part, vol-  
umes)  
*Get spatial interpolation volumes (intersection algorithm).*
- subroutine, private `cwp::cwp_field_intersection_tgt_elt_volumes_get_` (local\_code\_name, cpl\_id, field\_id, i\_↵  
\_part, tgt\_elt\_volumes)  
*Get spatial local target elements volumes (intersection algorithm).*
- subroutine, private `cwp::cwp_field_nearest_neighbors_distances_get_` (local\_code\_name, cpl\_id, field\_id,  
i\_part, distances2)  
*Get spatial interpolation distances (nearest neighbors algorithm).*
- subroutine, private `cwp::cwp_field_nearest_neighbors_coord_get_` (local\_code\_name, cpl\_id, field\_id, i\_part,  
nearest\_src\_coord)  
*Get coordinates of nearest source points (nearest neighbors algorithm).*
- subroutine, private `cwp::cwp_param_add_int_` (local\_code\_name, param\_name, initial\_value)  
*Add a new parameter and initialize it.*
- subroutine, private `cwp::cwp_param_add_double_` (local\_code\_name, param\_name, initial\_value)
- subroutine, private `cwp::cwp_param_add_char_` (local\_code\_name, param\_name, initial\_value)
- subroutine, private `cwp::cwp_param_set_int_` (local\_code\_name, param\_name, value)  
*Set a parameter.*
- subroutine, private `cwp::cwp_param_set_double_` (local\_code\_name, param\_name, value)
- subroutine, private `cwp::cwp_param_set_char_` (local\_code\_name, param\_name, value)
- subroutine, private `cwp::cwp_param_del_` (local\_code\_name, param\_name, data\_type)  
*Delete a parameter.*
- integer function, private `cwp::cwp_param_n_get_` (code\_name, data\_type)  
*Return the number of parameters for the code code\_name.*
- subroutine, private `cwp::cwp_param_list_get_` (code\_name, data\_type, n\_param, param\_names)  
*Return the list of parameters for the code code\_name.*
- integer function, private `cwp::cwp_param_is_` (code\_name, param\_name, data\_type)  
*Is this code\_name a parameter ?*
- subroutine, private `cwp::cwp_param_get_int` (code\_name, param\_name, value)  
*Return the parameter value of param\_name on code\_name.*
- subroutine, private `cwp::cwp_param_get_double` (code\_name, param\_name, value)
- subroutine, private `cwp::cwp_param_get_char` (code\_name, param\_name, val)
- subroutine, private `cwp::cwp_param_reduce_int` (op, param\_name, res, n\_codes, code\_names)  
*Return the result of a reduce operation about a parameter.*
- subroutine, private `cwp::cwp_param_reduce_double` (op, param\_name, res, n\_codes, code\_names)
- subroutine, private `cwp::cwp_param_reduce_char` (op, param\_name, res, n\_codes, code\_names)
- subroutine, private `cwp::cwp_param_lock_` (code\_name)  
*Lock access to local parameters from a distant code.*
- subroutine, private `cwp::cwp_param_unlock_` (code\_name)  
*Unlock access to local parameters from a distant code.*
- subroutine, private `cwp::cwp_global_data_issend_int` (local\_code\_name, cpl\_id, global\_data\_id, send\_data)

*Initiate the sending of a data array.*

- subroutine, private **cwp::cwp\_global\_data\_issend\_long** (local\_code\_name, cpl\_id, global\_data\_id, send\_data)
- subroutine, private **cwp::cwp\_global\_data\_issend\_double** (local\_code\_name, cpl\_id, global\_data\_id, send\_data)
- subroutine, private **cwp::cwp\_global\_data\_issend\_complex4** (local\_code\_name, cpl\_id, global\_data\_id, send\_data)
- subroutine, private **cwp::cwp\_global\_data\_issend\_complex8** (local\_code\_name, cpl\_id, global\_data\_id, send\_data)
- subroutine, private **cwp::cwp\_global\_data\_issend\_real4** (local\_code\_name, cpl\_id, global\_data\_id, send\_data)
- subroutine, private **cwp::cwp\_global\_data\_irecv\_int** (local\_code\_name, cpl\_id, global\_data\_id, recv\_data)

*Initiate the reception of a data array.*

- subroutine, private **cwp::cwp\_global\_data\_irecv\_long** (local\_code\_name, cpl\_id, global\_data\_id, recv\_data)
- subroutine, private **cwp::cwp\_global\_data\_irecv\_double** (local\_code\_name, cpl\_id, global\_data\_id, recv\_data)
- subroutine, private **cwp::cwp\_global\_data\_irecv\_complex4** (local\_code\_name, cpl\_id, global\_data\_id, recv\_data)
- subroutine, private **cwp::cwp\_global\_data\_irecv\_complex8** (local\_code\_name, cpl\_id, global\_data\_id, recv\_data)
- subroutine, private **cwp::cwp\_global\_data\_irecv\_real4** (local\_code\_name, cpl\_id, global\_data\_id, recv\_data)
- subroutine, private **cwp::cwp\_global\_data\_wait\_issend\_** (local\_code\_name, cpl\_id, global\_data\_id)

*Finalize the sending of a data array.*

- subroutine, private **cwp::cwp\_global\_data\_wait\_irecv\_** (local\_code\_name, cpl\_id, global\_data\_id)

*Finalize the reception of a data array.*

- subroutine, private **cwp::cwp\_part\_data\_create\_** (local\_code\_name, cpl\_id, part\_data\_id, exch\_type, gnum\_elt, n\_elt, n\_part)

*Create partitioned data exchange object.*

- subroutine, private **cwp::cwp\_part\_data\_del\_** (local\_code\_name, cpl\_id, part\_data\_id)

*Delete partitioned data exchange object.*

- subroutine, private **cwp::cwp\_part\_data\_issend\_** (local\_code\_name, cpl\_id, part\_data\_id, exch\_id, n\_components, send\_data)

*Send a data array.*

- subroutine, private **cwp::cwp\_part\_data\_irecv\_** (local\_code\_name, cpl\_id, part\_data\_id, exch\_id, n\_components, recv\_data)

*Receive a data array.*

- subroutine, private **cwp::cwp\_part\_data\_wait\_issend\_** (local\_code\_name, cpl\_id, part\_data\_id, exch\_id)

*Wait of send a data array.*

- subroutine, private **cwp::cwp\_part\_data\_wait\_irecv\_** (local\_code\_name, cpl\_id, part\_data\_id, exch\_id)

*Wait of receive a data array.*

- integer(c\_int) function, private **cwp::cwp\_cpl\_spatial\_interp\_algo\_get\_** (local\_code\_name, cpl\_id)

*Get the coupling spatial interpolation algorithm.*

### 4.1.1 Function/Subroutine Documentation

#### 4.1.1.1 cwp\_c\_to\_f\_string\_()

```
character(len=:)  function, pointer, private cwp::cwp_c_to_f_string_ (  
    character(kind=c_char,len=1), dimension(*), intent(in) c_str ) [private]
```

Create a Fortran string from a C string.

This function creates a Fortran string from a C string. There is a string copy

**Parameters**

in	<i>c_str</i>	C string
----	--------------	----------

**Returns**

Fortran string

**4.1.1.2 cwp\_codes\_list\_get\_()**

```
character(256) function, dimension(:), allocatable, private cwp::cwp_codes_list_get_ [private]
```

Return list of codes known by CWIPI.

**Returns**

List of code names

**4.1.1.3 cwp\_computed\_tgts\_bcast\_enable\_()**

```
subroutine, private cwp::cwp_computed_tgts_bcast_enable_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id ) [private]
```

Enable broadcast of the computed targets ids (in CWP\_COMM\_PAR\_WITHOUT\_PART mode).

This function must be called in order for the computed targets to be accessible on non-root ranks

**Parameters**

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Current partition

**4.1.1.4 cwp\_computed\_tgts\_get\_()**

```
integer(c_int) function, dimension(:), pointer, private cwp::cwp_computed_tgts_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part ) [private]
```

Return computed targets.



## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Current partition

## Returns

Computed targets

## 4.1.1.5 cwp\_cpl\_barrier\_()

```
subroutine, private cwp::cwp_cpl_barrier_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id ) [private]
```

MPI Barrier on the coupling communicator.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier

## 4.1.1.6 cwp\_cpl\_create\_()

```
subroutine, private cwp::cwp_cpl_create_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) coupled_code_name,
    integer(kind = c_int) entities_dim,
    integer(kind = c_int) comm_type,
    integer(kind = c_int) spatial_interp,
    integer(kind = c_int) n_part,
    integer(kind = c_int) displacement,
    integer(kind = c_int) freq ) [private]
```

Create a coupling object and define its properties.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>coupled_code_name</i>	Distant or local coupled code name
in	<i>comm_type</i>	Communication type
in	<i>spatial_interp</i>	Spatial interpolation method

**Parameters**

in	<i>n_part</i>	Number of interface partition
in	<i>displacement</i>	Mesh moving status
in	<i>recv_freq_type</i>	Type of receiving frequency

**4.1.1.7 cwp\_cpl\_del\_()**

```
subroutine, private cwp::cwp_cpl_del_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id ) [private]
```

Delete a coupling object.

**Parameters**

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier

**4.1.1.8 cwp\_cpl\_spatial\_interp\_algo\_get\_()**

```
integer(c_int) function, private cwp::cwp_cpl_spatial_interp_algo_get_ (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id ) [private]
```

Get the coupling spatial interpolation algorithm.

**Parameters**

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier

**Returns**

Spatial interpolation method

**4.1.1.9 cwp\_field\_create\_()**

```
subroutine, private cwp::cwp_field_create_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
```

```

character(kind = c_char, len = *) field_id,
integer(c_int) data_type,
integer(c_int) storage,
integer(c_int) n_component,
integer(c_int) target_location,
integer(c_int) exch_type,
integer(c_int) visu_status ) [private]

```

Create a new field.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field id
in	<i>data_type</i>	Data type
in	<i>storage</i>	Storage type
in	<i>n_component</i>	Number of component
in	<i>target_location</i>	Target location
in	<i>exch_type</i>	Exchange type
in	<i>visu_status</i>	Visualization status

#### 4.1.1.10 cwp\_field\_data\_set\_()

```

subroutine, private cwp::cwp_field_data_set_ (
character(kind = c_char, len = *) local_code_name,
character(kind = c_char, len = *) cpl_id,
character(kind = c_char, len = *) field_id,
integer(c_int) i_part,
integer(c_int) map_type,
double precision, dimension(:), pointer data ) [private]

```

Set field data.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Current partition
in	<i>data_type</i>	Choice if data is set for the source or the target
in	<i>data</i>	Storage array (Mapping)

#### 4.1.1.11 cwp\_field\_del\_()

```

subroutine, private cwp::cwp_field_del_ (
character(kind = c_char, len = *) local_code_name,

```

```
character(kind = c_char, len = *) cpl_id,
character(kind = c_char, len = *) field_id ) [private]
```

Delete a field.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier

#### 4.1.1.12 cwp\_field\_dof\_location\_get\_()

```
integer(c_int) function, private cwp::cwp_field_dof_location_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id ) [private]
```

Get target degrees of freedom location.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier

#### Returns

Location of degrees of freedom

#### 4.1.1.13 cwp\_field\_interp\_function\_set\_()

```
subroutine, private cwp::cwp_field_interp_function_set_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    user_interpolation_fct ) [private]
```

Setting of a user interpolation from location.

This function takes into account an user interpolation function written with void (\*CWP\_Field\_interp\_function\_t) interface.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>fct</i>	Function

**4.1.1.14 cwp\_field\_interp\_function\_unset\_()**

```

subroutine, private cwp::cwp_field_interp_function_unset_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id ) [private]

```

Unsetting of a user interpolation.

**Parameters**

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier

**4.1.1.15 cwp\_field\_intersection\_tgt\_elt\_volumes\_get\_()**

```

subroutine, private cwp::cwp_field_intersection_tgt_elt_volumes_get_ (
    character(kind=c_char, len = *) local_code_name,
    character(kind=c_char, len = *) cpl_id,
    character(kind=c_char, len = *) field_id,
    integer(c_int) i_part,
    double precision, dimension(:), pointer tgt_elt_volumes ) [private]

```

Get spatial local target elements volumes (intersection algorithm).

**Parameters**

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Partition identifier
out	<i>tgt_elt_volumes</i>	Volumes of local target elements

**4.1.1.16 cwp\_field\_intersection\_volumes\_get\_()**

```

subroutine, private cwp::cwp_field_intersection_volumes_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part,
    double precision, dimension(:), pointer volumes ) [private]

```

Get spatial interpolation volumes (intersection algorithm).

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Partition identifier
out	<i>volumes</i>	Volumes of intersection polyhedra

## 4.1.1.17 cwp\_field\_irecv\_()

```
subroutine, private cwp::cwp_field_irecv_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) tgt_field_id ) [private]
```

Receive a spatially interpolated field from the coupled code with non-blocking communications.

This function is independent of CWP\_Time\_exch\_t mode. The user has to manually check the consistency of the exchanges.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>tgt_field_id</i>	Target field id

## 4.1.1.18 cwp\_field\_issend\_()

```
subroutine, private cwp::cwp_field_issend_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id ) [private]
```

Send a spatially interpolated field to the coupled code with non-blocking communications.

This function is independent of CWP\_Time\_exch\_t mode. The user has to manually check the consistency of the exchanges.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier

**4.1.1.19 cwp\_field\_location\_internal\_cell\_vtx\_get\_()**

```

subroutine, private cwp::cwp_field_location_internal_cell_vtx_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part,
    integer(c_int), dimension(:), pointer cell_vtx_idx,
    integer(c_int), dimension(:), pointer cell_vtx ) [private]

```

Get spatial interpolation internal cell->vertex connectivity (location algorithm).

**Parameters**

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Partition identifier
out	<i>cell_vtx_idx</i>	Index for local cell->vertex connectivity
out	<i>cell_vtx</i>	Local cell->vertex connectivity

**4.1.1.20 cwp\_field\_location\_point\_data\_get\_()**

```

subroutine, private cwp::cwp_field_location_point_data_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part,
    double precision, dimension(:, :), pointer points_coords,
    double precision, dimension(:, :), pointer points_uvw,
    double precision, dimension(:), pointer points_dist2,
    double precision, dimension(:, :), pointer points_projected_coords ) [private]

```

Get spatial interpolation point data (location algorithm).

**Parameters**

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Partition identifier
out	<i>points_coords</i>	Cartesian coordinates of points inside local elements
out	<i>points_uvw</i>	Parametric coordinates of points inside local elements
out	<i>points_dist2</i>	Squared distance from points to elements
out	<i>points_projected_coords</i>	Cartesian coordinates of projection on points on local elements

#### 4.1.1.21 cwp\_field\_location\_weights\_get\_()

```
subroutine, private cwp::cwp_field_location_weights_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part,
    double precision, dimension(:), pointer weights ) [private]
```

Get spatial interpolation weights (location algorithm).

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Partition identifier
out	<i>weights</i>	Interpolation weights

#### 4.1.1.22 cwp\_field\_n\_components\_get\_()

```
integer function, private cwp::cwp_field_n_components_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id ) [private]
```

Get spatial interpolation number of algorithms.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier

#### 4.1.1.23 cwp\_field\_n\_dof\_get\_()

```
integer(c_int) function, private cwp::cwp_field_n_dof_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part ) [private]
```

Get field number of degrees of freedom.



## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>n_dof</i>	Field number of degrees of freedom

## Returns

Field storage type

## 4.1.1.24 cwp\_field\_nearest\_neighbors\_coord\_get\_()

```
subroutine, private cwp::cwp_field_nearest_neighbors_coord_get_ (
    character(kind=c_char, len = *) local_code_name,
    character(kind=c_char, len = *) cpl_id,
    character(kind=c_char, len = *) field_id,
    integer(c_int), intent(in) i_part,
    double precision, dimension(:,:), pointer nearest_src_coord ) [private]
```

Get coordinates of nearest source points (nearest neighbors algorithm).

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Partition identifier
out	<i>nearest_src_coord</i>	Coordinates of nearest source points

## 4.1.1.25 cwp\_field\_nearest\_neighbors\_distances\_get\_()

```
subroutine, private cwp::cwp_field_nearest_neighbors_distances_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part,
    double precision, dimension(:), pointer distances2 ) [private]
```

Get spatial interpolation distances (nearest neighbors algorithm).

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Partition identifier
out	<i>distances2</i>	Squared distances from nearest source points

#### 4.1.1.26 cwp\_field\_src\_data\_properties\_get\_()

```
subroutine, private cwp::cwp_field_src_data_properties_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part,
    integer(c_int) n_elt_src,
    integer(c_int), dimension(:), pointer src_to_tgt_idx ) [private]
```

Get spatial interpolation source data.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
out	<i>i_part</i>	Partition identifier
out	<i>n_elt_src</i>	Number of local source entities in current partition
out	<i>src_to_tgt_idx</i>	Index for source->target mapping

#### 4.1.1.27 cwp\_field\_storage\_get\_()

```
integer(c_int) function, private cwp::cwp_field_storage_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id ) [private]
```

Get field storage type.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier

##### Returns

Field storage type

#### 4.1.1.28 cwp\_field\_tgt\_data\_properties\_get\_()

```
subroutine, private cwp::cwp_field_tgt_data_properties_get_ (
    character(kind = c_char, len = *) local_code_name,
```

```

character(kind = c_char, len = *) cpl_id,
character(kind = c_char, len = *) field_id,
integer(c_int) i_part,
integer(c_int) n_elt_tgt,
integer(c_int) n_referenced_tgt,
integer(c_int), dimension(:), pointer referenced_tgt,
integer(c_int), dimension(:), pointer tgt_come_from_src_idx ) [private]

```

Get spatial interpolation target data.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
out	<i>i_part</i>	Partition identifier
out	<i>n_elt_tgt</i>	Number of local target entities in current partition
out	<i>n_referenced_tgt</i>	Number of referenced target entities in current partition
out	<i>referenced_tgt</i>	Ids of referenced target entities in current partition (1-based)
out	<i>tgt_come_from_src_idx</i>	Index for target->source mapping

#### 4.1.1.29 cwp\_field\_wait\_irecv\_()

```

subroutine, private cwp::cwp_field_wait_irecv_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) tgt_field_id ) [private]

```

Wait the end of an exchange related to request from CWP\_Field\_irecv.

This function waits the end of exchange related to request from CWP\_Field\_irecv

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>tgt_field_id</i>	Target field id

#### 4.1.1.30 cwp\_field\_wait\_issend\_()

```

subroutine, private cwp::cwp_field_wait_issend_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id ) [private]

```

Wait the end of an exchange related to request from CWP\_Field\_issend.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier

**4.1.1.31 cwp\_global\_data\_irecv\_int()**

```
subroutine, private cwp::cwp_global_data_irecv_int (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id,
    character(kind=c_char, len=*) global_data_id,
    integer(c_int), dimension(:, :), pointer recv_data ) [private]
```

Initiate the reception of a data array.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>global_data_id</i>	GlobalData identifier
in	<i>recv_data</i>	Pointer to data array

**4.1.1.32 cwp\_global\_data\_issend\_int()**

```
subroutine, private cwp::cwp_global_data_issend_int (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id,
    character(kind=c_char, len=*) global_data_id,
    integer(c_int), dimension(:, :), pointer send_data ) [private]
```

Initiate the sending of a data array.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>global_data_id</i>	GlobalData identifier
in	<i>send_data</i>	Pointer to data array

**4.1.1.33 cwp\_global\_data\_wait\_irecv\_()**

```
subroutine, private cwp::cwp_global_data_wait_irecv_ (
```

```

character(kind=c_char, len=*) local_code_name,
character(kind=c_char, len=*) cpl_id,
character(kind=c_char, len=*) global_data_id ) [private]

```

Finalize the reception of a data array.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>global_data_id</i>	GlobalData identifier

#### 4.1.1.34 cwp\_global\_data\_wait\_issend\_()

```

subroutine, private cwp::cwp_global_data_wait_issend_ (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id,
    character(kind=c_char, len=*) global_data_id ) [private]

```

Finalize the sending of a data array.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>global_data_id</i>	GlobalData identifier

#### 4.1.1.35 cwp\_init\_()

```

subroutine, private cwp::cwp_init_ (
    integer(c_int) fcomm,
    integer(c_int), intent(in) n_code,
    character(kind = c_char, len = *), dimension(n_code), target code_names,
    integer(c_int) is_active_rank,
    integer(c_int), dimension(:), pointer intra_comms ) [private]

```

Initialize CWIPI.

This function creates the MPI intra communicators of the codes from the `global_comm` MPI communicator that contains all code ranks. This function has to be called from all ranks contained in the `global_comm`.

#### Parameters

in	<i>global_comm</i>	MPI global communicator
in	<i>n_code</i>	Number of codes on the current rank
in	<i>code_names</i>	Names of codes on the current rank (size = <i>n_code</i> )
in	<i>is_active_rank</i>	Current rank is available for CWIPI
out	<i>intra_comms</i>	MPI intra communicators of each code (size = <i>n_code</i> )

#### 4.1.1.36 cwp\_involved\_srcs\_bcast\_enable\_()

```
subroutine, private cwp::cwp_involved_srcs_bcast_enable_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id ) [private]
```

Enable broadcast of the involved sources ids (in CWP\_COMM\_PAR\_WITHOUT\_PART mode).

This function must be called in order for the involved sources to be accessible on non-root ranks

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Current partition

#### 4.1.1.37 cwp\_involved\_srcs\_get\_()

```
integer(c_int) function, dimension(:), pointer, private cwp::cwp_involved_srcs_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part ) [private]
```

Return involved sources.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Current partition

##### Returns

Involved sources

#### 4.1.1.38 cwp\_loc\_codes\_list\_get\_()

```
character(256) function, dimension(:), allocatable, private cwp::cwp_loc_codes_list_get_↔
[private]
```

Return list of local codes known by CWIPI.

**Returns**

List of local code names

**4.1.1.39 cwp\_mesh\_interf\_block\_add\_()**

```
integer(c_int) function, private cwp::cwp_mesh_interf_block_add_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    integer(c_int) block_type ) [private]
```

Add a connectivity block to the interface mesh.

**Parameters**

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>block_type</i>	Block type

**Returns**

block identifier

**4.1.1.40 cwp\_mesh\_interf\_block\_std\_get\_()**

```
subroutine, private cwp::cwp_mesh_interf_block_std_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    integer(c_int) i_part,
    integer(c_int) block_id,
    integer(c_int) n_elts,
    integer(c_int), dimension(:), pointer connec,
    integer(c_long), dimension(:), pointer global_num ) [private]
```

Get the properties of a standard block of the interface mesh.

**Parameters**

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Partition identifier
in	<i>block_id</i>	Block identifier
out	<i>n_elts</i>	Number of elements
out	<i>connec</i>	Connectivity (size = n_vertex_elt * n_elts)
out	<i>global_num</i>	Pointer to global element number (or NULL)

#### 4.1.1.41 cwp\_mesh\_interf\_block\_std\_set\_()

```

subroutine, private cwp::cwp_mesh_interf_block_std_set_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    integer(c_int) i_part,
    integer(c_int) block_id,
    integer(c_int) n_elts,
    integer(c_int), dimension(:), pointer connec,
    integer(c_long), dimension(:), pointer global_num ) [private]

```

Set a standard block to the interface mesh.

This function adds a connectivity block to the interface mesh. Definition of element connectivity is :

- edge (CWP\_BLOCK\_EDGE2) :

```

1 x-----x 2

```

- triangle (CWP\_BLOCK\_FACE\_TRIA3):

```

1 x-----x 3
 \       /
  \     /
   \   /
    x 2

```

- quadrangle (CWP\_BLOCK\_FACE\_QUAD4) :

```

    4 x-----x 3
   /       \
  /         \
 /           \
1 x-----x2

```

- tetrahedron (CWP\_BLOCK\_CELL\_TETRA4) :

```

    x 4
   / \
  /   \
 /     \
1 x-----x 3
  \     /
   \   /
    x 2

```

- pyramid (CWP\_BLOCK\_CELL\_PYRAM5) :

```

    5 x
   / \
  /   \
 /     \
4 x-----x 3
  \     /
   \   /
    x 2

```

- prism (CWP\_BLOCK\_CELL\_PRISM6) :

```

4 x-----x 6
 |       |
 |       |
 |       |
1 x-----x 3
  \     /
   \   /
    x 2

```

- hexaedron (CWP\_BLOCK\_CELL\_HEX8) :

```

    8 x-----x 7
   / \       / \
  /   \     /   \
 /     \   /     \
5 x-----x6 |
 | 4x-----x 3
 | /       \ /
 | /       \ /
1 x-----x 2

```



## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Partition identifier
in	<i>block_id</i>	Block identifier
in	<i>n_elts</i>	Number of elements
in	<i>connec</i>	Connectivity (size = <i>n_vertex_elt</i> * <i>n_elts</i> )
in	<i>global_num</i>	Pointer to global element number (or NULL)

## 4.1.1.42 cwp\_mesh\_interf\_c\_poly\_block\_get\_()

```

subroutine, private cwp::cwp_mesh_interf_c_poly_block_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    integer(c_int) i_part,
    integer(c_int) block_id,
    integer(c_int) n_elts,
    integer(c_int) n_faces,
    integer(c_int), dimension(:), pointer connec_faces_idx,
    integer(c_int), dimension(:), pointer connec_faces,
    integer(c_int), dimension(:), pointer connec_cells_idx,
    integer(c_int), dimension(:), pointer connec_cells,
    integer(c_long), dimension(:), pointer global_num ) [private]

```

Get the properties of a polyhedron block of the interface mesh partition..

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Current partition
in	<i>block_id</i>	Block identifier
out	<i>n_elts</i>	Number of elements
out	<i>connec_cells_idx</i>	Polyhedron to face index ( <i>connec_cells_idx</i> [0] = 0 and size = <i>n_elts</i> + 1)
out	<i>connec_cells</i>	Polyhedron to face connectivity (size = <i>connec_cells_idx</i> [ <i>n_elts</i> ])
out	<i>n_faces</i>	Number of faces
out	<i>connec_faces_idx</i>	Polyhedron face to vertex index ( <i>connec_faces_idx</i> [0] = 0 and size = max( <i>cell_face_connec</i> ) + 1)
out	<i>connec_faces</i>	Polyhedron face to vertex connectivity (size = <i>connec_faces_idx</i> [ <i>n_elts</i> ])
out	<i>global_num</i>	Pointer to global element number (or NULL)

## 4.1.1.43 cwp\_mesh\_interf\_c\_poly\_block\_set\_()

```

subroutine, private cwp::cwp_mesh_interf_c_poly_block_set_ (
    character(kind = c_char, len = *) local_code_name,

```

```

character(kind = c_char, len = *) cpl_id,
integer(c_int) i_part,
integer(c_int) block_id,
integer(c_int) n_elts,
integer(c_int) n_faces,
integer(c_int), dimension(:), pointer connec_faces_idx,
integer(c_int), dimension(:), pointer connec_faces,
integer(c_int), dimension(:), pointer connec_cells_idx,
integer(c_int), dimension(:), pointer connec_cells,
integer(c_long), dimension(:), pointer global_num ) [private]

```

Adding a polyhedron connectivity block to the interface mesh.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Current partition
in	<i>block_id</i>	Block identifier
in	<i>n_elts</i>	Number of elements
in	<i>connec_cells_idx</i>	Polyhedron to face index (src_poly_cell_face_idx[0] = 0 and size = n_elts + 1)
in	<i>connec_cells</i>	Polyhedron to face connectivity (size = cell_face_idx[n_elts])
in	<i>n_faces</i>	Number of faces
in	<i>connec_faces_idx</i>	Polyhedron face to vertex index (connec_faces_idx[0] = 0 and size = max(cell_face_connec) + 1)
in	<i>connec_faces</i>	Polyhedron face to vertex connectivity (size = connec_faces_idx[n_elts])
in	<i>global_num</i>	Pointer to global element number (or NULL)

#### 4.1.1.44 cwp\_mesh\_interf\_del\_()

```

subroutine, private cwp::cwp_mesh_interf_del_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id ) [private]

```

Delete interface mesh.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier

#### 4.1.1.45 cwp\_mesh\_interf\_f\_poly\_block\_get\_()

```

subroutine, private cwp::cwp_mesh_interf_f_poly_block_get_ (
    character(kind = c_char, len = *) local_code_name,

```

```

character(kind = c_char, len = *) cpl_id,
integer(c_int) i_part,
integer(c_int) block_id,
integer(c_int) n_elts,
integer(c_int), dimension(:), pointer connec_idx,
integer(c_int), dimension(:), pointer connec,
integer(c_long), dimension(:), pointer global_num ) [private]

```

Get the properties of a polygon block of the interface mesh partition.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Current partition
in	<i>block_id</i>	Block identifier
out	<i>n_elts</i>	Number of elements
out	<i>connec_idx</i>	Connectivity index ( <i>connec_idx</i> [0] = 0 and size = <i>n_elts</i> + 1)
out	<i>connec</i>	Connectivity (size = <i>connec_idx</i> [ <i>n_elts</i> ])
out	<i>global_num</i>	Pointer to global element number (or NULL)

#### 4.1.1.46 cwp\_mesh\_interf\_f\_poly\_block\_set\_()

```

subroutine, private cwp::cwp_mesh_interf_f_poly_block_set_ (
character(kind = c_char, len = *) local_code_name,
character(kind = c_char, len = *) cpl_id,
integer(c_int) i_part,
integer(c_int) block_id,
integer(c_int) n_elts,
integer(c_int), dimension(:), pointer connec_idx,
integer(c_int), dimension(:), pointer connec,
integer(c_long), dimension(:), pointer global_num ) [private]

```

Set the connectivity of a polygon block in a interface mesh partition.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Current partition
in	<i>block_id</i>	Block identifier
in	<i>n_elts</i>	Number of elements
in	<i>connec_idx</i>	Connectivity index ( <i>connec_idx</i> [0] = 0 and size = <i>n_elts</i> + 1)
in	<i>connec</i>	Connectivity (size = <i>connec_idx</i> [ <i>n_elts</i> ])
in	<i>global_num</i>	Pointer to global element number (or NULL)

#### 4.1.1.47 cwp\_mesh\_interf\_finalize\_()

```
subroutine, private cwp::cwp_mesh_interf_finalize_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id ) [private]
```

Finalize interface mesh.

This function computes the global numbers of mesh entities if they are not provided.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier

#### 4.1.1.48 cwp\_mesh\_interf\_from\_cellface\_set\_()

```
subroutine, private cwp::cwp_mesh_interf_from_cellface_set_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    integer(c_int) i_part,
    integer(c_int) n_cells,
    integer(c_int), dimension(:), pointer cell_face_idx,
    integer(c_int), dimension(:), pointer cell_face,
    integer(c_int) n_faces,
    integer(c_int), dimension(:), pointer face_vtx_idx,
    integer(c_int), dimension(:), pointer face_vtx,
    integer(c_long), dimension(:), pointer global_num ) [private]
```

Define the interface mesh from a cell to face connectivity.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Current partition
in	<i>n_cells</i>	Number of cells
in	<i>cell_face_idx</i>	Polyhedron to face index (src_poly_cell_face_idx[0] = 0 and size = n_elts + 1)
in	<i>cell_face</i>	Cell to face connectivity (size = cell_face_idx[n_elts])
in	<i>n_faces</i>	Number of faces
in	<i>face_vtx_idx</i>	Polyhedron face to vertex index (face_vtx_idx[0] = 0 and size = n_faces + 1)
in	<i>face_vtx</i>	Face to vertex connectivity (size = face_vtx_idx[n_elts])
in	<i>global_num</i>	Pointer to parent element number (or NULL)

**4.1.1.49 cwp\_mesh\_intf\_from\_faceedge\_set\_()**

```

subroutine, private cwp::cwp_mesh_intf_from_faceedge_set_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    integer(c_int) i_part,
    integer(c_int) n_faces,
    integer(c_int), dimension(:), pointer face_edge_idx,
    integer(c_int), dimension(:), pointer face_edge,
    integer(c_int) n_edges,
    integer(c_int), dimension(:), pointer edge_vtx,
    integer(c_long), dimension(:), pointer global_num ) [private]

```

Define the surface interface mesh from a face to edge connectivity.

**Parameters**

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Current partition
in	<i>n_faces</i>	Number of cells
in	<i>face_edge_idx</i>	Polygon to edge index ( <i>face_edge_idx</i> [0] = 0 and size = <i>n_faces</i> + 1)
in	<i>face_edge</i>	Face to edge connectivity (size = <i>face_edge_idx</i> [ <i>n_faces</i> ])
in	<i>n_edges</i>	Number of faces
in	<i>edge_vtx</i>	Edge to vertex connectivity (size = 2 * <i>n_edges</i> )
in	<i>global_num</i>	Pointer to parent element number (or NULL)

**4.1.1.50 cwp\_mesh\_intf\_vtx\_set\_()**

```

subroutine, private cwp::cwp_mesh_intf_vtx_set_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    integer(c_int) i_part,
    integer(c_int) n_pts,
    double precision, dimension(:,:), pointer coord,
    integer(c_long), dimension(:), pointer global_num ) [private]

```

Set vertices.

**Parameters**

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Current partition
in	<i>n_pts</i>	Number of points
in	<i>coord</i>	Coordinates (size = 3 * <i>n_pts</i> )
in	<i>global_num</i>	Pointer to parent element number (or NULL)

#### 4.1.1.51 cwp\_n\_computed\_tgts\_get\_()

```
integer(c_int) function, private cwp::cwp_n_computed_tgts_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part ) [private]
```

Return the number of computed targets.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Current partition

##### Returns

Number of computed targets

#### 4.1.1.52 cwp\_n\_involved\_srcs\_get\_()

```
integer(c_int) function, private cwp::cwp_n_involved_srcs_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part ) [private]
```

Return the number of involved sources.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Current partition

##### Returns

Number of involved sources

#### 4.1.1.53 cwp\_n\_uncomputed\_tgts\_get\_()

```
integer(c_int) function, private cwp::cwp_n_uncomputed_tgts_get_ (
    character(kind = c_char, len = *) local_code_name,
```

```

character(kind = c_char, len = *) cpl_id,
character(kind = c_char, len = *) field_id,
integer(c_int) i_part ) [private]

```

Return the number of uncomputed targets.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Current partition

#### Returns

Number of uncomputed targets

#### 4.1.1.54 cwp\_output\_file\_set\_()

```

subroutine, private cwp::cwp_output_file_set_ (
    character(kind = c_char, len = *) f_output_file_name ) [private]

```

Define output file (in which only C code writes).

#### Parameters

in	<i>output_file_name</i>	Output file name
----	-------------------------	------------------

#### 4.1.1.55 cwp\_output\_fortran\_unit\_set()

```

subroutine cwp::cwp_output_fortran_unit_set (
    integer outputUnit )

```

Writing output to Fortran file (shared by fortran and C code).

This function set the file Fortran logical unit for writing output.

#### Parameters

in	<i>iunit</i>	File Fortan logical unit
----	--------------	--------------------------

#### 4.1.1.56 cwp\_param\_add\_int\_()

```
subroutine, private cwp::cwp_param_add_int_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) param_name,
    integer(kind = c_int), intent(in) initial_value ) [private]
```

Add a new parameter and initialize it.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>param_name</i>	Parameter name
in	<i>data_type</i>	Parameter type
in	<i>initial_value</i>	Initial value

#### 4.1.1.57 cwp\_param\_del\_()

```
subroutine, private cwp::cwp_param_del_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) param_name,
    integer, intent(in) data_type ) [private]
```

Delete a parameter.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>param_name</i>	Parameter name
in	<i>data_type</i>	Parameter type,

#### 4.1.1.58 cwp\_param\_get\_int()

```
subroutine, private cwp::cwp_param_get_int (
    character(kind = c_char, len = *) code_name,
    character(kind = c_char, len = *) param_name,
    integer(c_int), intent(out) value ) [private]
```

Return the parameter value of *param\_name* on *code\_name*.

##### Parameters

in	<i>code_name</i>	Local or distant code name
in	<i>param_name</i>	Parameter name
in	<i>data_type</i>	Parameter type
out	<i>value</i>	Parameter value



**4.1.1.59 cwp\_param\_is\_()**

```
integer function, private cwp::cwp_param_is_ (
    character(kind = c_char, len = *) code_name,
    character(kind = c_char, len = *) param_name,
    integer, intent(in) data_type ) [private]
```

Is this `code_name` a parameter ?

**Parameters**

in	<i>code_name</i>	Local or distant code name
in	<i>param_name</i>	Parameter name
in	<i>data_type</i>	Parameter type,

return 1 : true / 0 : false

**4.1.1.60 cwp\_param\_list\_get\_()**

```
subroutine, private cwp::cwp_param_list_get_ (
    character(kind = c_char, len = *) code_name,
    integer data_type,
    integer(c_int) n_param,
    character(256), dimension(:), allocatable param_names ) [private]
```

Return the list of parameters for the code `code_name`.

**Parameters**

in	<i>code_name</i>	Local or distant code name
in	<i>data_type</i>	Parameter type
in	<i>n_param</i>	Number of parameters
in	<i>param_names</i>	Parameter names

**4.1.1.61 cwp\_param\_lock\_()**

```
subroutine, private cwp::cwp_param_lock_ (
    character(kind = c_char, len = *) code_name ) [private]
```

Lock access to local parameters from a distant code.

**Parameters**

in	<i>code_name</i>	Code to lock
----	------------------	--------------

#### 4.1.1.62 cwp\_param\_n\_get\_()

```
integer function, private cwp::cwp_param_n_get_ (
    character(kind = c_char, len = *) code_name,
    integer, intent(in) data_type ) [private]
```

Return the number of parameters for the code `code_name`.

##### Parameters

in	<i>code_name</i>	Local or distant code name
in	<i>data_type</i>	Parameter type,

return Number of parameters

#### 4.1.1.63 cwp\_param\_reduce\_int()

```
subroutine, private cwp::cwp_param_reduce_int (
    integer, intent(in) op,
    character(kind = c_char, len = *) param_name,
    integer(c_int), intent(out) res,
    integer(c_int) n_codes,
    character(kind = c_char, len = *), dimension(n_codes), target code_names ) [private]
```

Return the result of a reduce operation about a parameter.

##### Parameters

in	<i>op</i>	Operation
in	<i>param_name</i>	Parameter name
in	<i>data_type</i>	Parameter type
out	<i>res</i>	Result
in	<i>n_codes</i>	Number of codes
in	<i>code_names</i>	Codes name

#### 4.1.1.64 cwp\_param\_set\_int\_()

```
subroutine, private cwp::cwp_param_set_int_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) param_name,
    integer(kind = c_int), intent(in) value ) [private]
```

Set a parameter.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>param_name</i>	Parameter name
in	<i>data_type</i>	Parameter type
in	<i>value</i>	Value

## 4.1.1.65 cwp\_param\_unlock\_()

```
subroutine, private cwp::cwp_param_unlock_ (
    character(kind = c_char, len = *) code_name ) [private]
```

Unlock access to local parameters from a distant code.

## Parameters

in	<i>code_name</i>	Code to unlock
----	------------------	----------------

## 4.1.1.66 cwp\_part\_data\_create\_()

```
subroutine, private cwp::cwp_part_data_create_ (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id,
    character(kind=c_char, len=*) part_data_id,
    integer(c_int), intent(in) exch_type,
    type(pdm_pointer_array_t), target gnum_elt,
    integer(c_int), dimension(:), pointer n_elt,
    integer, intent(in) n_part ) [private]
```

Create partitioned data exchange object.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>part_data_id</i>	PartData identifier
in	<i>exch_type</i>	Exchange type
in	<i>gnum_elt</i>	Global ids
in	<i>n_elt</i>	Number of elements in partitions (size = <i>n_part</i> )
in	<i>n_part</i>	Number of partitions

#### 4.1.1.67 cwp\_part\_data\_del\_()

```
subroutine, private cwp::cwp_part_data_del_ (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id,
    character(kind=c_char, len=*) part_data_id ) [private]
```

Delete partitioned data exchange object.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>part_data_id</i>	PartData identifier

#### 4.1.1.68 cwp\_part\_data\_irecv\_()

```
subroutine, private cwp::cwp_part_data_irecv_ (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id,
    character(kind=c_char, len=*) part_data_id,
    integer(c_int), intent(in) exch_id,
    integer(c_int), intent(in) n_components,
    type(pdm_pointer_array_t), target recv_data ) [private]
```

Receive a data array.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>part_data_id</i>	PartData identifier
in	<i>exch_id</i>	Exchange identifier
in	<i>n_components</i>	Number of components
in, out	<i>recv_data</i>	Pointer to data array to receive

#### 4.1.1.69 cwp\_part\_data\_issend\_()

```
subroutine, private cwp::cwp_part_data_issend_ (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id,
    character(kind=c_char, len=*) part_data_id,
    integer(c_int), intent(in) exch_id,
    integer(c_int), intent(in) n_components,
    type(pdm_pointer_array_t), target send_data ) [private]
```

Send a data array.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>part_data_id</i>	PartData identifier
in	<i>exch_id</i>	Exchange identifier
in	<i>n_components</i>	Number of components
in	<i>send_data</i>	Pointer to data array to send

## 4.1.1.70 cwp\_part\_data\_wait\_irecv\_()

```

subroutine, private cwp::cwp_part_data_wait_irecv_ (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id,
    character(kind=c_char, len=*) part_data_id,
    integer(c_int), intent(in) exch_id ) [private]

```

Wait of receive a data array.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>part_data_id</i>	PartData identifier
in	<i>exch_id</i>	Exchange identifier

## 4.1.1.71 cwp\_part\_data\_wait\_issend\_()

```

subroutine, private cwp::cwp_part_data_wait_issend_ (
    character(kind=c_char, len=*) local_code_name,
    character(kind=c_char, len=*) cpl_id,
    character(kind=c_char, len=*) part_data_id,
    integer(c_int), intent(in) exch_id ) [private]

```

Wait of send a data array.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>part_data_id</i>	PartData identifier
in	<i>exch_id</i>	Exchange identifier

#### 4.1.1.72 cwp\_spatial\_interp\_property\_set\_()

```
subroutine, private cwp::cwp_spatial_interp_property_set_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) property_name,
    integer (kind = c_int) property_type,
    character(kind = c_char, len = *) property_value ) [private]
```

Set a property of the spatial interpolation algorithm.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>property_name</i>	Name of the property
in	<i>property_type</i>	Type of the property
in	<i>property_value</i>	Value of the property

#### 4.1.1.73 cwp\_spatial\_interp\_weights\_compute\_()

```
subroutine, private cwp::cwp_spatial_interp_weights_compute_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id ) [private]
```

Compute spatial interpolation weights.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier

#### 4.1.1.74 cwp\_state\_get\_()

```
integer(c_int) function, private cwp::cwp_state_get_ (
    character(kind = c_char, len = *) local_code_name ) [private]
```

Return code state.

##### Parameters

in	<i>code_name</i>	Code name
----	------------------	-----------

## Returns

Code state

**4.1.1.75 cwp\_state\_update\_()**

```
subroutine, private cwp::cwp_state_update_ (  
    character(kind = c_char, len = *) local_code_name,  
    integer(kind = c_int), intent(in) state ) [private]
```

Update code state.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>state</i>	State

**4.1.1.76 cwp\_time\_step\_beg\_()**

```
subroutine, private cwp::cwp_time_step_beg_ (  
    character(kind = c_char, len = *) local_code_name,  
    double precision, intent(in) current_time ) [private]
```

Begin code time step.

## Parameters

in	<i>local_code_name</i>	Local code name
in	<i>current_time</i>	Current time

**4.1.1.77 cwp\_time\_step\_end\_()**

```
subroutine, private cwp::cwp_time_step_end_ (  
    character(kind = c_char, len = *) local_code_name ) [private]
```

End code time step.

## Parameters

in	<i>local_code_name</i>	Local code name
----	------------------------	-----------------

#### 4.1.1.78 cwp\_uncomputed\_tgts\_get\_()

```
integer(c_int) function, dimension(:), pointer, private cwp::cwp_uncomputed_tgts_get_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    character(kind = c_char, len = *) field_id,
    integer(c_int) i_part ) [private]
```

Return uncomputed targets.

##### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>field_id</i>	Field identifier
in	<i>i_part</i>	Current partition

##### Returns

Uncomputed targets

#### 4.1.1.79 cwp\_user\_structure\_get\_()

```
type(c_ptr) function, private cwp::cwp_user_structure_get_ (
    character(kind = c_char, len = *) local_code_name ) [private]
```

Return the user structure associated.

This structure can be called into a callback

##### Parameters

in	<i>local_code_name</i>	Local code name
----	------------------------	-----------------

##### Returns

User structure

#### 4.1.1.80 cwp\_user\_tgt\_pts\_set\_()

```
subroutine, private cwp::cwp_user_tgt_pts_set_ (
    character(kind = c_char, len = *) local_code_name,
    character(kind = c_char, len = *) cpl_id,
    integer(kind = c_int) i_part,
    integer(kind = c_int) n_pts,
```



```
double precision, dimension(:,:), pointer coord,
integer(kind = c_long), dimension(:), pointer global_num ) [private]
```

Setting user target points.

This function must be called if the degrees of freedom locations are CWP\_DOF\_LOCATION\_USER

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>i_part</i>	Current partition
in	<i>n_pts</i>	Number of points
in	<i>coord</i>	Coordinates (size = 3 * n_pts)
in	<i>g_num</i>	global number or NUL (size = n_pts)

#### 4.1.1.81 cwp\_visu\_set\_()

```
subroutine, private cwp::cwp_visu_set_ (
character(kind = c_char, len = *) local_code_name,
character(kind = c_char, len = *) cpl_id,
integer(c_int) freq,
integer(c_int) format,
character(kind = c_char, len = *) format_option ) [private]
```

Enable visualization output.

#### Parameters

in	<i>local_code_name</i>	Local code name
in	<i>cpl_id</i>	Coupling identifier
in	<i>freq</i>	Output frequency
in	<i>format</i>	Output format to visualize exchanged fieldsDouble on the coupled mesh. Choice between : <ul style="list-style-type: none"> <li>• "EnSight Gold"</li> </ul>
in	<i>format_option</i>	Output options "opt1, opt2, ..." <ul style="list-style-type: none"> <li>• text : output text files</li> <li>• binary : output binary files (default)</li> </ul>



# Index

cwp::cwp\_c\_to\_f\_string, [5](#)  
    cwp\_c\_to\_f\_string\_, [5](#)  
cwp::cwp\_codes\_list\_get, [6](#)  
    cwp\_codes\_list\_get\_, [6](#)  
cwp::CWP\_Codes\_nb\_get, [6](#)  
cwp::cwp\_computed\_tgt\_bcast\_enable, [7](#)  
    cwp\_computed\_tgt\_bcast\_enable\_, [7](#)  
cwp::cwp\_computed\_tgt\_get, [7](#)  
    cwp\_computed\_tgt\_get\_, [7](#)  
cwp::cwp\_cpl\_barrier, [8](#)  
    cwp\_cpl\_barrier\_, [8](#)  
cwp::cwp\_cpl\_create, [9](#)  
    cwp\_cpl\_create\_, [9](#)  
cwp::cwp\_cpl\_del, [10](#)  
    cwp\_cpl\_del\_, [10](#)  
cwp::cwp\_cpl\_spatial\_interp\_algo\_get, [10](#)  
    cwp\_cpl\_spatial\_interp\_algo\_get\_, [10](#)  
cwp::cwp\_field\_create, [11](#)  
    cwp\_field\_create\_, [11](#)  
cwp::cwp\_field\_data\_set, [12](#)  
    cwp\_field\_data\_set\_, [12](#)  
cwp::cwp\_field\_del, [13](#)  
    cwp\_field\_del\_, [13](#)  
cwp::cwp\_field\_dof\_location\_get, [13](#)  
    cwp\_field\_dof\_location\_get\_, [13](#)  
cwp::cwp\_field\_interp\_function\_set, [14](#)  
    cwp\_field\_interp\_function\_set\_, [14](#)  
cwp::cwp\_field\_interp\_function\_unset, [15](#)  
    cwp\_field\_interp\_function\_unset\_, [15](#)  
cwp::cwp\_field\_intersection\_tgt\_elt\_volumes\_get, [16](#)  
    cwp\_field\_intersection\_tgt\_elt\_volumes\_get\_, [16](#)  
cwp::cwp\_field\_intersection\_volumes\_get, [16](#)  
    cwp\_field\_intersection\_volumes\_get\_, [16](#)  
cwp::cwp\_field\_irecv, [17](#)  
    cwp\_field\_irecv\_, [17](#)  
cwp::cwp\_field\_issend, [18](#)  
    cwp\_field\_issend\_, [18](#)  
cwp::cwp\_field\_location\_internal\_cell\_vtx\_get, [19](#)  
    cwp\_field\_location\_internal\_cell\_vtx\_get\_, [19](#)  
cwp::cwp\_field\_location\_point\_data\_get, [19](#)  
    cwp\_field\_location\_point\_data\_get\_, [20](#)  
cwp::cwp\_field\_location\_weights\_get, [20](#)  
    cwp\_field\_location\_weights\_get\_, [20](#)  
cwp::cwp\_field\_n\_components\_get, [21](#)  
    cwp\_field\_n\_components\_get\_, [21](#)  
cwp::cwp\_field\_n\_dof\_get, [22](#)  
    cwp\_field\_n\_dof\_get\_, [22](#)  
cwp::cwp\_field\_nearest\_neighbors\_coord\_get, [22](#)  
    cwp\_field\_nearest\_neighbors\_coord\_get\_, [23](#)  
cwp::cwp\_field\_nearest\_neighbors\_distances\_get, [23](#)  
    cwp\_field\_nearest\_neighbors\_distances\_get\_, [23](#)  
cwp::cwp\_field\_src\_data\_properties\_get, [24](#)  
    cwp\_field\_src\_data\_properties\_get\_, [24](#)  
cwp::cwp\_field\_storage\_get, [25](#)  
    cwp\_field\_storage\_get\_, [25](#)  
cwp::cwp\_field\_tgt\_data\_properties\_get, [25](#)  
    cwp\_field\_tgt\_data\_properties\_get\_, [26](#)  
cwp::cwp\_field\_wait\_irecv, [26](#)  
    cwp\_field\_wait\_irecv\_, [26](#)  
cwp::cwp\_field\_wait\_issend, [27](#)  
    cwp\_field\_wait\_issend\_, [27](#)  
cwp::CWP\_Finalize, [28](#)  
cwp::cwp\_global\_data\_irecv, [28](#)  
    cwp\_global\_data\_irecv\_int, [28](#)  
cwp::cwp\_global\_data\_issend, [29](#)  
    cwp\_global\_data\_issend\_int, [29](#)  
cwp::cwp\_global\_data\_wait\_irecv, [30](#)  
    cwp\_global\_data\_wait\_irecv\_, [30](#)  
cwp::cwp\_global\_data\_wait\_issend, [30](#)  
    cwp\_global\_data\_wait\_issend\_, [30](#)  
cwp::cwp\_init, [31](#)  
    cwp\_init\_, [31](#)  
cwp::cwp\_involved\_srcs\_bcast\_enable, [32](#)  
    cwp\_involved\_srcs\_bcast\_enable\_, [32](#)  
cwp::cwp\_involved\_srcs\_get, [32](#)  
    cwp\_involved\_srcs\_get\_, [32](#)  
cwp::cwp\_loc\_codes\_list\_get, [33](#)  
    cwp\_loc\_codes\_list\_get\_, [33](#)  
cwp::CWP\_Loc\_codes\_nb\_get, [34](#)  
cwp::cwp\_mesh\_interf\_block\_add, [34](#)  
    cwp\_mesh\_interf\_block\_add\_, [34](#)  
cwp::cwp\_mesh\_interf\_block\_std\_get, [35](#)  
    cwp\_mesh\_interf\_block\_std\_get\_, [35](#)  
cwp::cwp\_mesh\_interf\_block\_std\_set, [36](#)  
    cwp\_mesh\_interf\_block\_std\_set\_, [36](#)  
cwp::cwp\_mesh\_interf\_c\_poly\_block\_get, [37](#)  
    cwp\_mesh\_interf\_c\_poly\_block\_get\_, [37](#)  
cwp::cwp\_mesh\_interf\_c\_poly\_block\_set, [38](#)  
    cwp\_mesh\_interf\_c\_poly\_block\_set\_, [38](#)  
cwp::cwp\_mesh\_interf\_del, [39](#)  
    cwp\_mesh\_interf\_del\_, [39](#)  
cwp::cwp\_mesh\_interf\_f\_poly\_block\_get, [40](#)  
    cwp\_mesh\_interf\_f\_poly\_block\_get\_, [40](#)  
cwp::cwp\_mesh\_interf\_f\_poly\_block\_set, [41](#)  
    cwp\_mesh\_interf\_f\_poly\_block\_set\_, [41](#)  
cwp::cwp\_mesh\_interf\_finalize, [42](#)  
    cwp\_mesh\_interf\_finalize\_, [42](#)  
cwp::cwp\_mesh\_interf\_from\_cellface\_set, [42](#)

- cwp\_mesh\_intf\_from\_cellface\_set\_, 42
- cwp::cwp\_mesh\_intf\_from\_faceedge\_set\_, 43
  - cwp\_mesh\_intf\_from\_faceedge\_set\_, 43
- cwp::cwp\_mesh\_intf\_vtx\_set\_, 44
  - cwp\_mesh\_intf\_vtx\_set\_, 44
- cwp::cwp\_n\_computed\_tgts\_get\_, 45
  - cwp\_n\_computed\_tgts\_get\_, 45
- cwp::cwp\_n\_involved\_srcs\_get\_, 46
  - cwp\_n\_involved\_srcs\_get\_, 46
- cwp::cwp\_n\_uncomputed\_tgts\_get\_, 46
  - cwp\_n\_uncomputed\_tgts\_get\_, 47
- cwp::cwp\_output\_file\_set\_, 47
  - cwp\_output\_file\_set\_, 47
- cwp::cwp\_param\_add\_, 48
  - cwp\_param\_add\_int\_, 48
- cwp::cwp\_param\_del\_, 48
  - cwp\_param\_del\_, 49
- cwp::cwp\_param\_get\_, 49
  - cwp\_param\_get\_int\_, 49
- cwp::cwp\_param\_is\_, 50
  - cwp\_param\_is\_, 50
- cwp::cwp\_param\_list\_get\_, 51
  - cwp\_param\_list\_get\_, 51
- cwp::cwp\_param\_lock\_, 51
  - cwp\_param\_lock\_, 51
- cwp::cwp\_param\_n\_get\_, 52
  - cwp\_param\_n\_get\_, 52
- cwp::cwp\_param\_reduce\_, 52
  - cwp\_param\_reduce\_int\_, 53
- cwp::cwp\_param\_set\_, 53
  - cwp\_param\_set\_int\_, 53
- cwp::cwp\_param\_unlock\_, 54
  - cwp\_param\_unlock\_, 54
- cwp::cwp\_part\_data\_create\_, 54
  - cwp\_part\_data\_create\_, 55
- cwp::cwp\_part\_data\_del\_, 55
  - cwp\_part\_data\_del\_, 55
- cwp::cwp\_part\_data\_irecv\_, 56
  - cwp\_part\_data\_irecv\_, 56
- cwp::cwp\_part\_data\_issend\_, 57
  - cwp\_part\_data\_issend\_, 57
- cwp::cwp\_part\_data\_wait\_irecv\_, 57
  - cwp\_part\_data\_wait\_irecv\_, 57
- cwp::cwp\_part\_data\_wait\_issend\_, 58
  - cwp\_part\_data\_wait\_issend\_, 58
- cwp::CWP\_Properties\_dump\_, 59
- cwp::cwp\_spatial\_interp\_property\_set\_, 59
  - cwp\_spatial\_interp\_property\_set\_, 59
- cwp::cwp\_spatial\_interp\_weights\_compute\_, 60
  - cwp\_spatial\_interp\_weights\_compute\_, 60
- cwp::cwp\_state\_get\_, 60
  - cwp\_state\_get\_, 61
- cwp::cwp\_state\_update\_, 61
  - cwp\_state\_update\_, 61
- cwp::cwp\_time\_step\_beg\_, 62
  - cwp\_time\_step\_beg\_, 62
- cwp::cwp\_time\_step\_end\_, 62
  - cwp\_time\_step\_end\_, 62
- cwp::cwp\_uncomputed\_tgts\_get\_, 63
  - cwp\_uncomputed\_tgts\_get\_, 63
- cwp::cwp\_user\_structure\_get\_, 64
  - cwp\_user\_structure\_get\_, 64
- cwp::cwp\_user\_structure\_set\_, 64
- cwp::cwp\_user\_tgt\_pts\_set\_, 65
  - cwp\_user\_tgt\_pts\_set\_, 65
- cwp::cwp Visu\_set\_, 65
  - cwp Visu\_set\_, 66
- cwp\_c\_to\_f\_string\_
  - cwp::cwp\_c\_to\_f\_string\_, 5
  - cwp\_f.f90, 73
- cwp\_codes\_list\_get\_
  - cwp::cwp\_codes\_list\_get\_, 6
  - cwp\_f.f90, 75
- cwp\_computed\_tgts\_bcast\_enable\_
  - cwp::cwp\_computed\_tgts\_bcast\_enable\_, 7
  - cwp\_f.f90, 75
- cwp\_computed\_tgts\_get\_
  - cwp::cwp\_computed\_tgts\_get\_, 7
  - cwp\_f.f90, 75
- cwp\_cpl\_barrier\_
  - cwp::cwp\_cpl\_barrier\_, 8
  - cwp\_f.f90, 77
- cwp\_cpl\_create\_
  - cwp::cwp\_cpl\_create\_, 9
  - cwp\_f.f90, 77
- cwp\_cpl\_del\_
  - cwp::cwp\_cpl\_del\_, 10
  - cwp\_f.f90, 78
- cwp\_cpl\_spatial\_interp\_algo\_get\_
  - cwp::cwp\_cpl\_spatial\_interp\_algo\_get\_, 10
  - cwp\_f.f90, 78
- cwp\_f.f90
  - cwp\_c\_to\_f\_string\_, 73
  - cwp\_codes\_list\_get\_, 75
  - cwp\_computed\_tgts\_bcast\_enable\_, 75
  - cwp\_computed\_tgts\_get\_, 75
  - cwp\_cpl\_barrier\_, 77
  - cwp\_cpl\_create\_, 77
  - cwp\_cpl\_del\_, 78
  - cwp\_cpl\_spatial\_interp\_algo\_get\_, 78
  - cwp\_field\_create\_, 78
  - cwp\_field\_data\_set\_, 79
  - cwp\_field\_del\_, 79
  - cwp\_field\_dof\_location\_get\_, 80
  - cwp\_field\_interp\_function\_set\_, 80
  - cwp\_field\_interp\_function\_unset\_, 81
  - cwp\_field\_intersection\_tgt\_elt\_volumes\_get\_, 81
  - cwp\_field\_intersection\_volumes\_get\_, 81
  - cwp\_field\_irecv\_, 82
  - cwp\_field\_issend\_, 82
  - cwp\_field\_location\_internal\_cell\_vtx\_get\_, 82
  - cwp\_field\_location\_point\_data\_get\_, 83
  - cwp\_field\_location\_weights\_get\_, 83
  - cwp\_field\_n\_components\_get\_, 84
  - cwp\_field\_n\_dof\_get\_, 84
  - cwp\_field\_nearest\_neighbors\_coord\_get\_, 85

- cwp\_field\_nearest\_neighbors\_distances\_get\_, 85
- cwp\_field\_src\_data\_properties\_get\_, 86
- cwp\_field\_storage\_get\_, 86
- cwp\_field\_tgt\_data\_properties\_get\_, 86
- cwp\_field\_wait\_irecv\_, 87
- cwp\_field\_wait\_issend\_, 87
- cwp\_global\_data\_irecv\_int, 88
- cwp\_global\_data\_issend\_int, 88
- cwp\_global\_data\_wait\_irecv\_, 88
- cwp\_global\_data\_wait\_issend\_, 89
- cwp\_init\_, 89
- cwp\_involved\_srcs\_bcast\_enable\_, 90
- cwp\_involved\_srcs\_get\_, 90
- cwp\_loc\_codes\_list\_get\_, 90
- cwp\_mesh\_interf\_block\_add\_, 91
- cwp\_mesh\_interf\_block\_std\_get\_, 91
- cwp\_mesh\_interf\_block\_std\_set\_, 92
- cwp\_mesh\_interf\_c\_poly\_block\_get\_, 93
- cwp\_mesh\_interf\_c\_poly\_block\_set\_, 93
- cwp\_mesh\_interf\_del\_, 94
- cwp\_mesh\_interf\_f\_poly\_block\_get\_, 94
- cwp\_mesh\_interf\_f\_poly\_block\_set\_, 95
- cwp\_mesh\_interf\_finalize\_, 95
- cwp\_mesh\_interf\_from\_cellface\_set\_, 96
- cwp\_mesh\_interf\_from\_faceedge\_set\_, 96
- cwp\_mesh\_interf\_vtx\_set\_, 97
- cwp\_n\_computed\_tgts\_get\_, 97
- cwp\_n\_involved\_srcs\_get\_, 98
- cwp\_n\_uncomputed\_tgts\_get\_, 98
- cwp\_output\_file\_set\_, 99
- cwp\_output\_fortran\_unit\_set, 99
- cwp\_param\_add\_int\_, 99
- cwp\_param\_del\_, 100
- cwp\_param\_get\_int, 100
- cwp\_param\_is\_, 101
- cwp\_param\_list\_get\_, 101
- cwp\_param\_lock\_, 101
- cwp\_param\_n\_get\_, 102
- cwp\_param\_reduce\_int, 102
- cwp\_param\_set\_int\_, 102
- cwp\_param\_unlock\_, 103
- cwp\_part\_data\_create\_, 103
- cwp\_part\_data\_del\_, 103
- cwp\_part\_data\_irecv\_, 104
- cwp\_part\_data\_issend\_, 104
- cwp\_part\_data\_wait\_irecv\_, 105
- cwp\_part\_data\_wait\_issend\_, 105
- cwp\_spatial\_interp\_property\_set\_, 105
- cwp\_spatial\_interp\_weights\_compute\_, 106
- cwp\_state\_get\_, 106
- cwp\_state\_update\_, 107
- cwp\_time\_step\_beg\_, 107
- cwp\_time\_step\_end\_, 107
- cwp\_uncomputed\_tgts\_get\_, 107
- cwp\_user\_structure\_get\_, 108
- cwp\_user\_tgt\_pts\_set\_, 108
- cwp\_visu\_set\_, 109
- cwp\_field\_create\_
  - cwp::cwp\_field\_create, 11
  - cwp\_f.f90, 78
- cwp\_field\_data\_set\_
  - cwp::cwp\_field\_data\_set, 12
  - cwp\_f.f90, 79
- cwp\_field\_del\_
  - cwp::cwp\_field\_del, 13
  - cwp\_f.f90, 79
- cwp\_field\_dof\_location\_get\_
  - cwp::cwp\_field\_dof\_location\_get, 13
  - cwp\_f.f90, 80
- cwp\_field\_interp\_function\_set\_
  - cwp::cwp\_field\_interp\_function\_set, 14
  - cwp\_f.f90, 80
- cwp\_field\_interp\_function\_unset\_
  - cwp::cwp\_field\_interp\_function\_unset, 15
  - cwp\_f.f90, 81
- cwp\_field\_intersection\_tgt\_elt\_volumes\_get\_
  - cwp::cwp\_field\_intersection\_tgt\_elt\_volumes\_get, 16
  - cwp\_f.f90, 81
- cwp\_field\_intersection\_volumes\_get\_
  - cwp::cwp\_field\_intersection\_volumes\_get, 16
  - cwp\_f.f90, 81
- cwp\_field\_irecv\_
  - cwp::cwp\_field\_irecv, 17
  - cwp\_f.f90, 82
- cwp\_field\_issend\_
  - cwp::cwp\_field\_issend, 18
  - cwp\_f.f90, 82
- cwp\_field\_location\_internal\_cell\_vtx\_get\_
  - cwp::cwp\_field\_location\_internal\_cell\_vtx\_get, 19
  - cwp\_f.f90, 82
- cwp\_field\_location\_point\_data\_get\_
  - cwp::cwp\_field\_location\_point\_data\_get, 20
  - cwp\_f.f90, 83
- cwp\_field\_location\_weights\_get\_
  - cwp::cwp\_field\_location\_weights\_get, 20
  - cwp\_f.f90, 83
- cwp\_field\_n\_components\_get\_
  - cwp::cwp\_field\_n\_components\_get, 21
  - cwp\_f.f90, 84
- cwp\_field\_n\_dof\_get\_
  - cwp::cwp\_field\_n\_dof\_get, 22
  - cwp\_f.f90, 84
- cwp\_field\_nearest\_neighbors\_coord\_get\_
  - cwp::cwp\_field\_nearest\_neighbors\_coord\_get, 23
  - cwp\_f.f90, 85
- cwp\_field\_nearest\_neighbors\_distances\_get\_
  - cwp::cwp\_field\_nearest\_neighbors\_distances\_get, 23
  - cwp\_f.f90, 85
- cwp\_field\_src\_data\_properties\_get\_
  - cwp::cwp\_field\_src\_data\_properties\_get, 24
  - cwp\_f.f90, 86
- cwp\_field\_storage\_get\_
  - cwp::cwp\_field\_storage\_get, 25
  - cwp\_f.f90, 86

cwp\_field\_tgt\_data\_properties\_get\_  
     cwp::cwp\_field\_tgt\_data\_properties\_get, 26  
     cwp\_f.f90, 86  
 cwp\_field\_wait\_irecv\_  
     cwp::cwp\_field\_wait\_irecv, 26  
     cwp\_f.f90, 87  
 cwp\_field\_wait\_issend\_  
     cwp::cwp\_field\_wait\_issend, 27  
     cwp\_f.f90, 87  
 cwp\_global\_data\_irecv\_int  
     cwp::cwp\_global\_data\_irecv, 28  
     cwp\_f.f90, 88  
 cwp\_global\_data\_issend\_int  
     cwp::cwp\_global\_data\_issend, 29  
     cwp\_f.f90, 88  
 cwp\_global\_data\_wait\_irecv\_  
     cwp::cwp\_global\_data\_wait\_irecv, 30  
     cwp\_f.f90, 88  
 cwp\_global\_data\_wait\_issend\_  
     cwp::cwp\_global\_data\_wait\_issend, 30  
     cwp\_f.f90, 89  
 cwp\_init\_  
     cwp::cwp\_init, 31  
     cwp\_f.f90, 89  
 cwp\_involved\_srcs\_bcast\_enable\_  
     cwp::cwp\_involved\_srcs\_bcast\_enable, 32  
     cwp\_f.f90, 90  
 cwp\_involved\_srcs\_get\_  
     cwp::cwp\_involved\_srcs\_get, 32  
     cwp\_f.f90, 90  
 cwp\_loc\_codes\_list\_get\_  
     cwp::cwp\_loc\_codes\_list\_get, 33  
     cwp\_f.f90, 90  
 cwp\_mesh\_interf\_block\_add\_  
     cwp::cwp\_mesh\_interf\_block\_add, 34  
     cwp\_f.f90, 91  
 cwp\_mesh\_interf\_block\_std\_get\_  
     cwp::cwp\_mesh\_interf\_block\_std\_get, 35  
     cwp\_f.f90, 91  
 cwp\_mesh\_interf\_block\_std\_set\_  
     cwp::cwp\_mesh\_interf\_block\_std\_set, 36  
     cwp\_f.f90, 92  
 cwp\_mesh\_interf\_c\_poly\_block\_get\_  
     cwp::cwp\_mesh\_interf\_c\_poly\_block\_get, 37  
     cwp\_f.f90, 93  
 cwp\_mesh\_interf\_c\_poly\_block\_set\_  
     cwp::cwp\_mesh\_interf\_c\_poly\_block\_set, 38  
     cwp\_f.f90, 93  
 cwp\_mesh\_interf\_del\_  
     cwp::cwp\_mesh\_interf\_del, 39  
     cwp\_f.f90, 94  
 cwp\_mesh\_interf\_f\_poly\_block\_get\_  
     cwp::cwp\_mesh\_interf\_f\_poly\_block\_get, 40  
     cwp\_f.f90, 94  
 cwp\_mesh\_interf\_f\_poly\_block\_set\_  
     cwp::cwp\_mesh\_interf\_f\_poly\_block\_set, 41  
     cwp\_f.f90, 95  
 cwp\_mesh\_interf\_finalize\_  
     cwp::cwp\_mesh\_interf\_finalize, 42  
     cwp\_f.f90, 95  
 cwp\_mesh\_interf\_from\_cellface\_set\_  
     cwp::cwp\_mesh\_interf\_from\_cellface\_set, 42  
     cwp\_f.f90, 96  
 cwp\_mesh\_interf\_from\_faceedge\_set\_  
     cwp::cwp\_mesh\_interf\_from\_faceedge\_set, 43  
     cwp\_f.f90, 96  
 cwp\_mesh\_interf\_vtx\_set\_  
     cwp::cwp\_mesh\_interf\_vtx\_set, 44  
     cwp\_f.f90, 97  
 cwp\_n\_computed\_tgts\_get\_  
     cwp::cwp\_n\_computed\_tgts\_get, 45  
     cwp\_f.f90, 97  
 cwp\_n\_involved\_srcs\_get\_  
     cwp::cwp\_n\_involved\_srcs\_get, 46  
     cwp\_f.f90, 98  
 cwp\_n\_uncomputed\_tgts\_get\_  
     cwp::cwp\_n\_uncomputed\_tgts\_get, 47  
     cwp\_f.f90, 98  
 cwp\_output\_file\_set\_  
     cwp::cwp\_output\_file\_set, 47  
     cwp\_f.f90, 99  
 cwp\_output\_fortran\_unit\_set\_  
     cwp\_f.f90, 99  
 cwp\_param\_add\_int\_  
     cwp::cwp\_param\_add, 48  
     cwp\_f.f90, 99  
 cwp\_param\_del\_  
     cwp::cwp\_param\_del, 49  
     cwp\_f.f90, 100  
 cwp\_param\_get\_int\_  
     cwp::cwp\_param\_get, 49  
     cwp\_f.f90, 100  
 cwp\_param\_is\_  
     cwp::cwp\_param\_is, 50  
     cwp\_f.f90, 101  
 cwp\_param\_list\_get\_  
     cwp::cwp\_param\_list\_get, 51  
     cwp\_f.f90, 101  
 cwp\_param\_lock\_  
     cwp::cwp\_param\_lock, 51  
     cwp\_f.f90, 101  
 cwp\_param\_n\_get\_  
     cwp::cwp\_param\_n\_get, 52  
     cwp\_f.f90, 102  
 cwp\_param\_reduce\_int\_  
     cwp::cwp\_param\_reduce, 53  
     cwp\_f.f90, 102  
 cwp\_param\_set\_int\_  
     cwp::cwp\_param\_set, 53  
     cwp\_f.f90, 102  
 cwp\_param\_unlock\_  
     cwp::cwp\_param\_unlock, 54  
     cwp\_f.f90, 103  
 cwp\_part\_data\_create\_  
     cwp::cwp\_part\_data\_create, 55  
     cwp\_f.f90, 103

- cwp\_part\_data\_del\_
  - cwp::cwp\_part\_data\_del, [55](#)
  - cwp\_f.f90, [103](#)
- cwp\_part\_data\_irecv\_
  - cwp::cwp\_part\_data\_irecv, [56](#)
  - cwp\_f.f90, [104](#)
- cwp\_part\_data\_issend\_
  - cwp::cwp\_part\_data\_issend, [57](#)
  - cwp\_f.f90, [104](#)
- cwp\_part\_data\_wait\_irecv\_
  - cwp::cwp\_part\_data\_wait\_irecv, [57](#)
  - cwp\_f.f90, [105](#)
- cwp\_part\_data\_wait\_issend\_
  - cwp::cwp\_part\_data\_wait\_issend, [58](#)
  - cwp\_f.f90, [105](#)
- cwp\_spatial\_interp\_property\_set\_
  - cwp::cwp\_spatial\_interp\_property\_set, [59](#)
  - cwp\_f.f90, [105](#)
- cwp\_spatial\_interp\_weights\_compute\_
  - cwp::cwp\_spatial\_interp\_weights\_compute, [60](#)
  - cwp\_f.f90, [106](#)
- cwp\_state\_get\_
  - cwp::cwp\_state\_get, [61](#)
  - cwp\_f.f90, [106](#)
- cwp\_state\_update\_
  - cwp::cwp\_state\_update, [61](#)
  - cwp\_f.f90, [107](#)
- cwp\_time\_step\_beg\_
  - cwp::cwp\_time\_step\_beg, [62](#)
  - cwp\_f.f90, [107](#)
- cwp\_time\_step\_end\_
  - cwp::cwp\_time\_step\_end, [62](#)
  - cwp\_f.f90, [107](#)
- cwp\_uncomputed\_tgts\_get\_
  - cwp::cwp\_uncomputed\_tgts\_get, [63](#)
  - cwp\_f.f90, [107](#)
- cwp\_user\_structure\_get\_
  - cwp::cwp\_user\_structure\_get, [64](#)
  - cwp\_f.f90, [108](#)
- cwp\_user\_tgt\_pts\_set\_
  - cwp::cwp\_user\_tgt\_pts\_set, [65](#)
  - cwp\_f.f90, [108](#)
- cwp\_visu\_set\_
  - cwp::cwp\_visu\_set, [66](#)
  - cwp\_f.f90, [109](#)
- fortran/new/cwp\_f.f90, [67](#)