Appendix D.
Structure Definitions

Topic: Healing

This appendix contains the definitions of some structures that are used by some HEAL classes. The structures are listed in alphabetical order by structure name.

bhl_advanced_spline_solver_results

Topic: Healing

```
// Structure for Advanced Spline Solver_results

struct DECL_HEAL bhl_advanced_spline_solver_results {

    int bhl_no_nets;                    // # faces to be netted
    int bhl_tgt_spl_no_nets_4sided;     // # 4-sided patches made
    int bhl_tgt_spl_no_nets_3sided;     // # 3-sided patches made
    int bhl_tgt_spl_no_bad_nets;        // # junction failures
    int bhl_tgt_spl_unsolvable;         // # unsolvable junctions

    // Constructor function
    bhl_advanced_spline_solver_results()
    {
        bhl_no_nets = 0;
        bhl_tgt_spl_no_nets_4sided = 0;
        bhl_tgt_spl_no_nets_3sided = 0;
        bhl_tgt_spl_no_bad_nets = 0;
        bhl_tgt_spl_unsolvable = 0;
    }
};
```

bhl_analytic_solver_results

Topic: Healing

```
// Structure for Analytic Solver Results

struct DECL_HEAL bhl_analytic_solver_results

{
    int bhl_tangents_resolved;      // # Analytic tangent junctions resolved
    int bhl_tangents_unresolved;    // # Analytic tangent junctions unresolved
    int bhl_intersections_resolved; // # Analytic intersections resolved
    int bhl_intersections_unresolved;   // # Analytic intersections unresolved
    int bhl_vertices_resolved;      // # vertices resolved
    int bhl_vertices_unresolved;    // # vertices unresolved
```

```
    // Breakup
    int edges_intersected;  // # edges calculated by analytic intersections
    int edges_exact_projected;  // # edges calculated by exact projections
    int edges_approx_projected; // # edges calculated by approx projections
    int coincident_snaps;       // # coincident snappings resolved
    int vertices_intersected;   // # vertices calculated by intersections
    int vertices_projected;     // # vertices calculated by projections
    int unstable_vertices_corrected;    // # unstable vertices corrected

    int degree_of_graph;        // Degree of the snapper graph
    int bodies_reversed;        // # bodies reversed.

    // Constructor
    bhl_analytic_solver_results()
    {
        bhl_tangents_resolved = 0;
        bhl_tangents_unresolved = 0;
        bhl_intersections_resolved = 0;
        bhl_intersections_unresolved = 0;
        bhl_vertices_resolved = 0;
        bhl_vertices_unresolved = 0;
        edges_intersected = 0;
        edges_exact_projected = 0;
        edges_approx_projected = 0;
        coincident_snaps = 0;
        vertices_intersected = 0;
        vertices_projected = 0;
        unstable_vertices_corrected = 0;
        degree_of_graph = 0;
        bodies_reversed = 0;
    }
};
```

bhl_anal_geometry_results

Topic: Healing

```
// Structure for Geometry Analysis

struct DECL_HEAL bhl_anal_geometry_results {

    int num_total_edges;        // # edges
    int num_bad_edges;          // # bad edges
    int num_total_coedges;      // # coedges
    int num_bad_coedges;        // # bad coedges
    int num_total_vertices;     // # vertices
    int num_bad_vertices;       // # bad vertices

    int num_analytic_transverse_edges;  // # bad transverse analytic junctions
    int num_spline_transverse_edges;    // # bad transverse spline junctions
```

```
    int num_tangent_edges;              // # bad tangent junctions
    int num_tangent_edges_analytic;     // # bad analytic tangents
    int num_G1_analytic_tangent_edges;  //# G1 bad analytic tangent edges
    int num_tangent_edges_uv_uv;        // # bad tangent uvsplines
    int num_tangent_edges_uv_nonuv;     // # bad tangent uv-nouv junctions
    int num_tangent_edges_nonuv_nonuv;  // # bad tangent nonuv-nouv junctions
    int num_tangent_edges_3_4_sided;    // # bad tangent splines with 3 or 4
                                        // sided faces on either sides

    int num_tangent_uv_uv_boun_boun;    // # bad tangent uv boundary-uv
                                        // junctions
    int num_tangent_complete_uv_uv;     // # bad tangent complete uv boundary
                                        // junctions
    int num_tangent_uv_uv_boun_internal;    // # bad tangent uv boundary-uv
                                        // internal junctions
    int num_tangent_uv_uv_internal_internal;    // # bad tangent uv
                                            // internal-uv internal junctions


    int num_bad_poles;              // # poles
    int num_total_surfaces;         // # bad surfaces
    int num_bad_surfaces;           // # discontinuous surfaces
    logical all_edge_vert_good;     // Are all the edges/vertices good?
    logical all_geom_good;          // Are all geometries good
    int healed_percentage;          // Percentage of geometries good
    logical within_scope_of_healer; // Whether all bad geometry is within the
                                    // scope of body-healer


    // multithreading
    int bhl_anal_geom_checked_surfaces; // # surfaces checked
    int bhl_anal_geom_checked_edges;    // # edges checked
    int bhl_anal_geom_checked_coedges;  // # coedges checked
    int bhl_anal_geom_checked_vertices; // # vertices checked

// Constructor function
    bhl_anal_geometry_results ()
    {
        num_total_edges = 0;
        num_bad_edges = 0;
        num_total_coedges = 0;
        num_bad_coedges = 0;
        num_total_vertices = 0;
        num_bad_vertices = 0;

        num_analytic_transverse_edges = 0;
        num_spline_transverse_edges = 0;
```

```
        num_tangent_edges = 0;
        num_tangent_edges_analytic = 0;
        num_G1_analytic_tangent_edges = 0;
        num_tangent_edges_uv_uv = 0;
        num_tangent_edges_uv_nonuv = 0;
        num_tangent_edges_nonuv_nonuv = 0;
        num_tangent_edges_3_4_sided = 0;

        num_tangent_uv_uv_boun_boun = 0;
        num_tangent_complete_uv_uv = 0;
        num_tangent_uv_uv_boun_internal = 0;
        num_tangent_uv_uv_internal_internal = 0;

        num_bad_poles = 0;
        num_total_surfaces = 0;
        num_bad_surfaces = 0;
        healed_percentage = 0;
        all_edge_vert_good = TRUE;
        all_geom_good = TRUE;
        within_scope_of_healer = TRUE;

        bhl_anal_geom_checked_surfaces= 0;
        bhl_anal_geom_checked_edges = 0;
        bhl_anal_geom_checked_coedges = 0;
        bhl_anal_geom_checked_vertices= 0;
    }
};
```

bhl_anal_stitch_results

Topic: Healing

```
// Structure for Stitch Analysis

struct DECL_HEAL bhl_anal_stitch_results {

    ANAL_STITCH bhl_anal_stitch;            // Denotes input analysis summary
    double  bhl_stitch_recommended_min_tol; // Recommended minimum tolerance
    double  bhl_stitch_recommended_max_tol; // Recommended maximum tolerance
    logical bhl_ignore_topology_recommended;    // Whether incoming topology
                                                // be ignored

    // Results topology check of input body
    bhl_topology_check_results topology_check_results;

    // Constructor function
    bhl_anal_stitch_results ()
    {
        bhl_stitch_recommended_min_tol = 0.0;
        bhl_stitch_recommended_min_tol = 0.0;
        bhl_ignore_topology_recommended = TRUE;
    }
};
```

bhl_geombld_options

Topic: Healing

```
struct bhl_geombld_options
{
    int bhl_geom_bld_approx_proj;
    int bhl_geom_bld_proj_vert;
    int bhl_geom_bld_good_vert;
    int bhl_geom_bld_good_intersect;
    int bhl_geom_bld_bad_intersect;

    bhl_geombld_options()
    {
        bhl_geom_bld_approx_proj = 0;
        bhl_geom_bld_proj_vert = 0;
        bhl_geom_bld_good_vert = 0;
        bhl_geom_bld_good_intersect = 0;
        bhl_geom_bld_bad_intersect = 0;
    }
};
```

bhl_geometry_results

Topic: Healing

```
// Structure for Geometry results

struct DECL_HEAL bhl_geometry_results {

    // Results of analytic solver
    struct bhl_analytic_solver_results analytic_solver_results;
    // Results of spline solver
    struct bhl_spline_solver_results spline_solver_results;
    // Results of transversal solver
    struct bhl_transversal_solver_results transversal_solver_results;
    // Results of advanced spline solver
    struct bhl_advanced_spline_solver_results advanced_spline_solver_results;
    // Results of wrapup
    struct bhl_wrapup_results wrapup_results;
    // Results of reblending
    struct bhl_reblend_results reblend_results;
};
```

bhl_geom_misc

Topic: Healing

```
struct DECL_HEAL bhl_geom_misc
{
    double  bhl_sim_doub_tol;
    double  bhl_sim_nor_tol;
    double  bhl_sim_pos_tol;
    int bhl_closed_split;
    int bhl_sim_no_ell;
    int bhl_sim_no_cir;
    int bhl_sim_no_str;
    int bhl_sim_processed_faces;
};
```

bhl_geom_types

Topic: Healing

```
// Structure containing the number of each type of geometry

struct bhl_geom_types {

    int no_splines;     // # splines in the entities
    int no_analytics;   // # analytics in the entities
    int no_planes;      // # planes in the entities
    int no_cylinders;   // # cylinders in the entities
    int no_cones;       // # cones in the entities
    int no_tori;        // # tori in the entities
    int no_spheres;     // # spheres in the entities

    //Constructor
    bhl_geom_types()
    {
        no_splines = 0;
        no_analytics = 0;
        no_planes = 0;
        no_cylinders = 0;
        no_cones = 0;
        no_tori = 0;
        no_spheres = 0;
    }
};
```

bhl_spline_solver_results

Topic: Healing

```
// Structure for Isospline Solver Results
```

```
struct DECL_HEAL bhl_spline_solver_results {

    int n_isospline_edges_present;  // # isospline tangent junctions in body
    int edges_resolved;             // # isospline tangent junctions resolved
    int edges_unresolved;           // # isospline tangent junctions
                                    // unresolved

    int n_complete_range_present;   // # complete range isospline junctions
    int n_subset_present;           // # subset range isospline junctions
    int n_overlap_present;          // # overlap range isospline junctions
    int n_spline_plane_present;     // # spline-plane junctions in the body
    int n_spline_analytic_present;  // # spline-analytic junctions in the body

    int n_complete_range_resolved;  // # complete range isospline junctions
                                    // resolved
    int n_subset_resolved;          // # subset range isospline junctions
                                    // resolved
    int n_overlap_resolved;         // # overlap range isospline junctions
                                    // resolved
    int n_spline_plane_resolved;    // # spline-plane junctions resolved
    int n_spline_analytic_resolved; // # spline-analytic junctions resolved

    int n_edges_made_g1;            // # edges made g1 successfully
    int n_edges_failed_g1;          // # edges failed to make g1
    int n_edges_unsolvable_g1;      // # edges not within current scope for g1
    int n_iso_splines_bent;         // # splines bent at vertices
    int n_iso_spline_bending_failed;    // # failures in spline bending

    // Constructor
    bhl_spline_solver_results() {init();}
    void init()
    {
        n_isospline_edges_present = 0;

        edges_resolved = 0;
        edges_unresolved = 0;

        n_complete_range_present = 0;
        n_subset_present = 0;
        n_overlap_present = 0;
        n_spline_plane_present = 0;
        n_spline_analytic_present = 0;

        n_complete_range_resolved = 0;
        n_subset_resolved = 0;
        n_overlap_resolved = 0;
        n_spline_plane_resolved = 0;
        n_spline_analytic_resolved = 0;
```

```
            n_edges_made_g1 = 0;
            n_edges_failed_g1 = 0;
            n_edges_unsolvable_g1 = 0;

            n_iso_splines_bent = 0;
            n_iso_spline_bending_failed = 0;
        }
    };
```

bhl_stitch_options

Topic: Healing

```
    struct DECL_HEAL bhl_stitch_options {

    double bhl_curr_stitch_tol ;
        logical bhl_ignore_top ;
        double bhl_stitch_min_tol ;
        double bhl_stitch_max_tol ;
        int bhl_stitch_steps ;
        logical bhl_stitch_check_normals ;
        logical bhl_stitch_face_normals ;
        logical bhl_make_one_solid;
        logical hh_stitch_attrib ;
        logical bhl_stitch ;
        logical bhl_stitch_plus;
        logical bhl_stitch_option;  //TRUE if incremental
        double bhl_body_box_size;
        double bhl_tang_tol;


        //constructor function
        bhl_stitch_options()
        {
            bhl_curr_stitch_tol = 0;
            bhl_ignore_top = FALSE;
            bhl_stitch_min_tol = 0.00001;
            bhl_stitch_max_tol = 1.0;
            bhl_stitch_steps = 4;
            bhl_stitch_check_normals = FALSE;
            bhl_stitch_face_normals = FALSE;
            bhl_make_one_solid = FALSE;
            hh_stitch_attrib = FALSE;
            bhl_stitch = TRUE;
            bhl_stitch_plus = FALSE;
            bhl_stitch_option = TRUE;
            bhl_body_box_size = -1.0;
        }
    };
```

bhl_stitch_results

Topic: Healing

```
// Structure for Stitch results

struct DECL_HEAL bhl_stitch_results {

    int bhl_stitch_no_solids;              // # solid bodies made
    int bhl_stitch_no_open;                // # open bodies made
    int bhl_stitch_no_unshared_edges;      // # final unshared edges
    int bhl_no_unshared_loops;             // # unshared loops
    int bhl_no_valid_unshared_edges;       // # valid unshared edges
    int bhl_no_invalid_unshared_edges;     // # invalid unshared edges
    int bhl_stitch_curr_shared_edges;
    int bhl_stitch_curr_total_edges;
    int bhl_stitch_no_total_edges;
    int bhl_stitch_tot_unshared_edges;
    int bhl_stitch_num_solids;             // # solid bodies made
    int bhl_stitch_num_open;               // # solid bodies made
    int bhl_stitch_curr_tol_count;
    int bhl_stitch_curr_tol_order;
    int bhl_tot_no_coalasced_edges;
    int bhl_tot_no_merged_edges;
    int bhl_tot_no_split_edges;
    double bhl_stitch_tol;
    logical bhl_curr_stitch_state;

    // Results topology check of stitched body
    bhl_topology_check_results topology_check_results;
```

```
    // Constructor function
    bhl_stitch_results ()
    {
        bhl_stitch_no_solids = 0;
        bhl_stitch_no_open = 0;
        bhl_stitch_no_unshared_edges = 0;
        bhl_no_unshared_loops = 0;
        bhl_no_valid_unshared_edges = 0;
        bhl_no_invalid_unshared_edges = 0;
        bhl_stitch_curr_shared_edges = 0;
        bhl_stitch_curr_total_edges = 0;
        bhl_stitch_no_total_edges = 0;
        bhl_stitch_tot_unshared_edges = 0;
        bhl_stitch_num_solids = 0;
        bhl_stitch_num_open = 0;
        bhl_stitch_curr_tol_count = 0;
        bhl_stitch_curr_tol_order = 0;
        bhl_tot_no_coalasced_edges = 0;
        bhl_tot_no_merged_edges = 0;
        bhl_tot_no_split_edges = 0;
        bhl_stitch_tol = 0.01;
        bhl_curr_stitch_state = 0;
    }
};
```

bhl_transversal_solver_results

Topic: Healing

```
// Structure for Transversal Solver Results

struct DECL_HEAL bhl_transversal_solver_results {

    int bhl_edges_resolved;         // # edges resolved
    int bhl_edges_unresolved;       // # edges unresolved
    int bhl_vertices_resolved;      // # vertices resolved
    int bhl_vertices_unresolved;    // # vertices unresolved

    // Breakup
    int edges_intersected;          // # edges resolved by intersections
    int edges_exact_projected;      // # edges resolved by exact projections
    int edges_approx_projected;     // # edges resolved by approx projections
    int vertices_intersected;       // # vertices resolved by intersections
    int vertices_exact_projected;   // # vertices resolved by exact
                                    // projections
    int vertices_approx_projected;  // # vertices resolved by approx
                                    // projections
```

```
        // Constructor
        bhl_transversal_solver_results()
        {
            bhl_edges_resolved = 0;
            bhl_edges_unresolved = 0;
            bhl_vertices_resolved = 0;
            bhl_vertices_unresolved = 0;
            edges_intersected = 0;
            edges_exact_projected = 0;
            edges_approx_projected = 0;
            vertices_intersected = 0;
            vertices_exact_projected = 0;
            vertices_approx_projected = 0;
        }
    };
```

bhl_wrapup_results

Topic: Healing

```
    // Struct for wrapup results

    struct DECL_HEAL bhl_wrapup_results {

        int pcurves_computed;       // # pcurves computed
        int edges_trimmed;          // # edges trimmed

        // Constructor function
        bhl_wrapup_results()
        {
            pcurves_computed = 0;
            edges_trimmed = 0;
        }
    };
```

hh_advspl_options

Topic: Healing

```
    struct hh_advspl_options {
        logical use_advspl;
        hh_advspl_options();
    };
```

hh_anal_solv_options

Topic: Healing

```
    struct hh_anal_solv_options {
```

```
        logical bhl_anal_tgt;
        double bhl_anal_tgt_tol;
        double bhl_snap_nor_tol;
        double bhl_scale_tol;

        hh_anal_solv_options()
        {   bhl_anal_tgt = TRUE;
            bhl_anal_tgt_tol = 0.01;
            bhl_snap_nor_tol = 0.001;
            bhl_scale_tol = 0.01;
        }
    };
```

hh_coedge_details

Topic: Healing

```
    struct DECL_HEAL hh_coedge_details {

        PAR_LINE partype;
        double parval;
        PAR_LOC parloc;
        logical end_to_end;
        logical rev;

        int face_type;      // identity of face (SPLINE_TYPE, PLANE_TYPE, etc..)
        int n_sided_face;

        // Constructor
        hh_coedge_details()
        {
            init();
        }

        void init();
        void reset_param_details();

        logical is_uv();
        logical is_uv_boun();
        logical is_end_to_end();
        logical is_analytic();
    };
```

hh_geombuild_options

Topic: Healing

```
    // Geometry building options
```

```
struct hh_geombuild_options {
    double min_geombuild_tol;
    double  geombuild_tol;
    logical use_geombuild;
    logical check_discontinuity;
    double bhl_tang_tol;
    double hh_min_spline_tang_tol;
    double hh_max_spline_tang_tol;

    hh_geombuild_options ();
    void reset();
    void set();
};
```

hh_isospline_options

Topic: Healing

```
// Isospline module options.

struct hh_isospline_options {
    logical use_spline_tgt;
    logical make_isospline_c1;
    logical do_c1_twist_correction;

    hh_isospline_options();
    void set();
    void reset();
};
```

hh_secondary_solver_options

Topic: Healing

```
struct hh_secondary_solver_options {
    logical use_secondary_solver;
    logical bhl_wrapup;

    hh_secondary_solver_options () {
        use_secondary_solver = TRUE;
        bhl_wrapup = TRUE;
    }

    void reset() {use_secondary_solver = bhl_wrapup;}
    void set() {bhl_wrapup = use_secondary_solver;}
}
```

hh_sharped_options

Topic: Healing

```
struct hh_sharped_options {
    logical fix_geometry;
```

```
    hh_sharped_options();
    void set();
    void reset();
};
```