

```

inline void setup_fields( grit::engine2d_type & engine, grit::param_type & parameters)
{
    engine.attributes().create_attribute( "lower_field", 1u);
    engine.attributes().create_attribute( "upper_field", 1u);

    glue::clear_attribute( engine, "lower_field", lower_value, glue::EDGE_ATTRIBUTE() );
    glue::clear_attribute( engine, "upper_field", upper_value, glue::EDGE_ATTRIBUTE() );

    parameters.set_lower_threshold_attribute( "refinement", "lower_field");
    parameters.set_lower_threshold_attribute( "interface_refinement", "lower_field");

    parameters.set_upper_threshold_attribute( "coarsening", "upper_field");
    parameters.set_upper_threshold_attribute( "interface_coarsening", "upper_field");
}

```

Step 1: Connect custom edge fields to threshold limits on named operations

```

inline void do_simulation_step(
    grit::engine2d_type & engine
    , util::ConfigFile const & settings
)
{
    std::vector<double> L;
    std::vector<double> U;

    glue::Phase const my_area = glue::make_phase(engine,...);

    glue::get_sub_range(engine, my_area, "lower_field", L, glue::EDGE_ATTRIBUTE());
    glue::get_sub_range(engine, my_area, "upper_field", U, glue::EDGE_ATTRIBUTE());

    //... Update L and U with new values

    glue::set_sub_range(engine, my_area, "lower_field", L, glue::EDGE_ATTRIBUTE());
    glue::set_sub_range(engine, my_area, "upper_field", U, glue::EDGE_ATTRIBUTE());
}

```

Step 2: Set the field values using glue::get_sub_range & glue::set_sub_range. GRIT takes care of everything else.

If sizing fields are not specified for all attributes GRIT is just going to default back to the default value set with glue::clear_attribute or if glue::glue_attribute is omitted then the lower/upper threshold value that is read from the cfg files are used as default value.

How to set and control GRIT parameters in cfg files