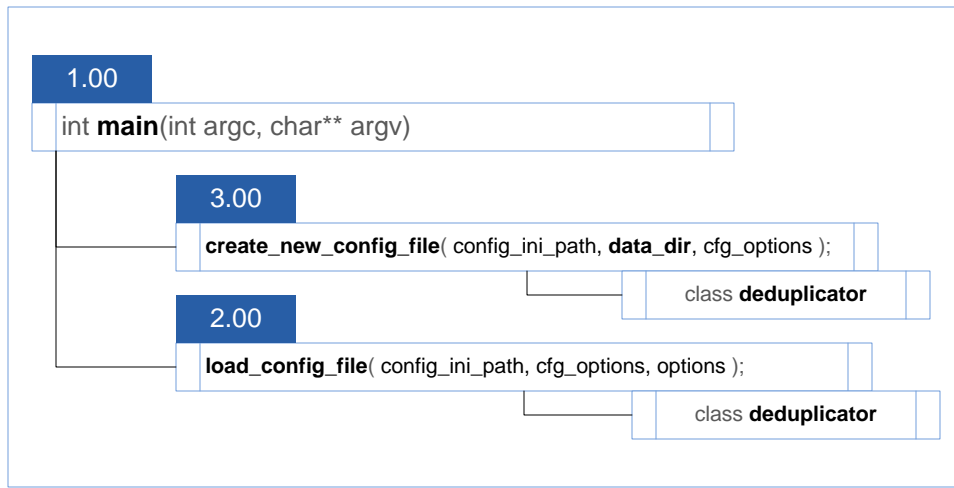


witness_node starting functions structures and the code flow



```
using namespace graphene;
```

```
namespace bpo = boost::program_options;
```

```
void write_default_logging_config_to_stream(std::ostream& out);
```

```
fc::optional<fc::logging_config> load_logging_config_from_ini_file(const fc::path& config_ini_filename);
```

```
class deduplicator
```

4.00

```
public:
```

```
deduplicator() : modifier(nullptr) {}
```

```
deduplicator(const boost::shared_ptr<bpo::option_description> (*mod_fn)(const
boost::shared_ptr<bpo::option_description>&))
: modifier(mod_fn) {}
```

```
const boost::shared_ptr<bpo::option_description> next(const boost::shared_ptr<bpo::option_description>& o)
{
    const std::string name = o->long_name();
    if (seen.find( name ) != seen.end() )
        return nullptr;
    seen.insert(name);
    return modifier ? modifier(o) : o;
}
```

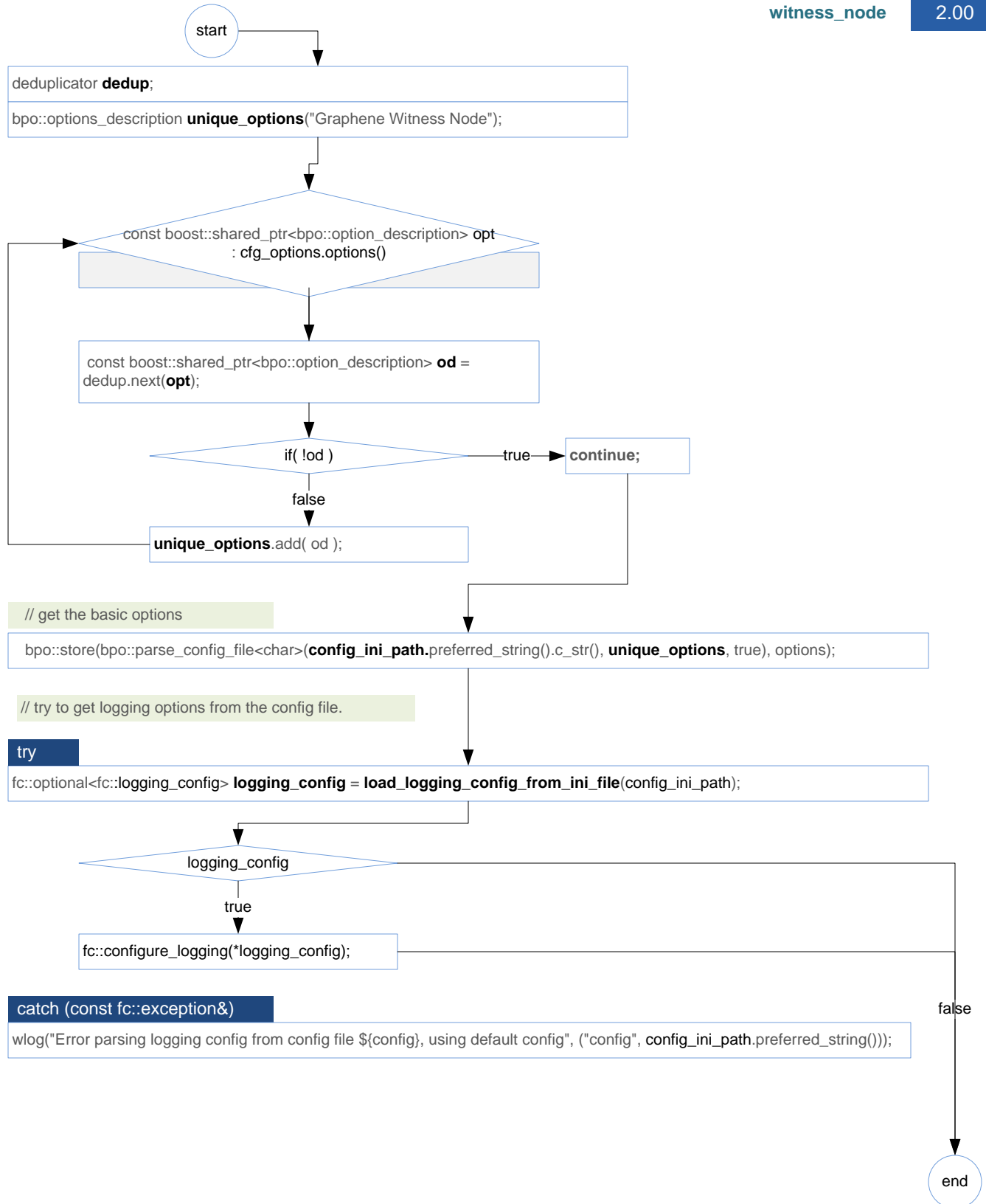
```
private:
```

```
boost::container::flat_set<std::string> seen;
const boost::shared_ptr<bpo::option_description> (*modifier)(const boost::shared_ptr<bpo::option_description>&);
```

witness_node

2.00

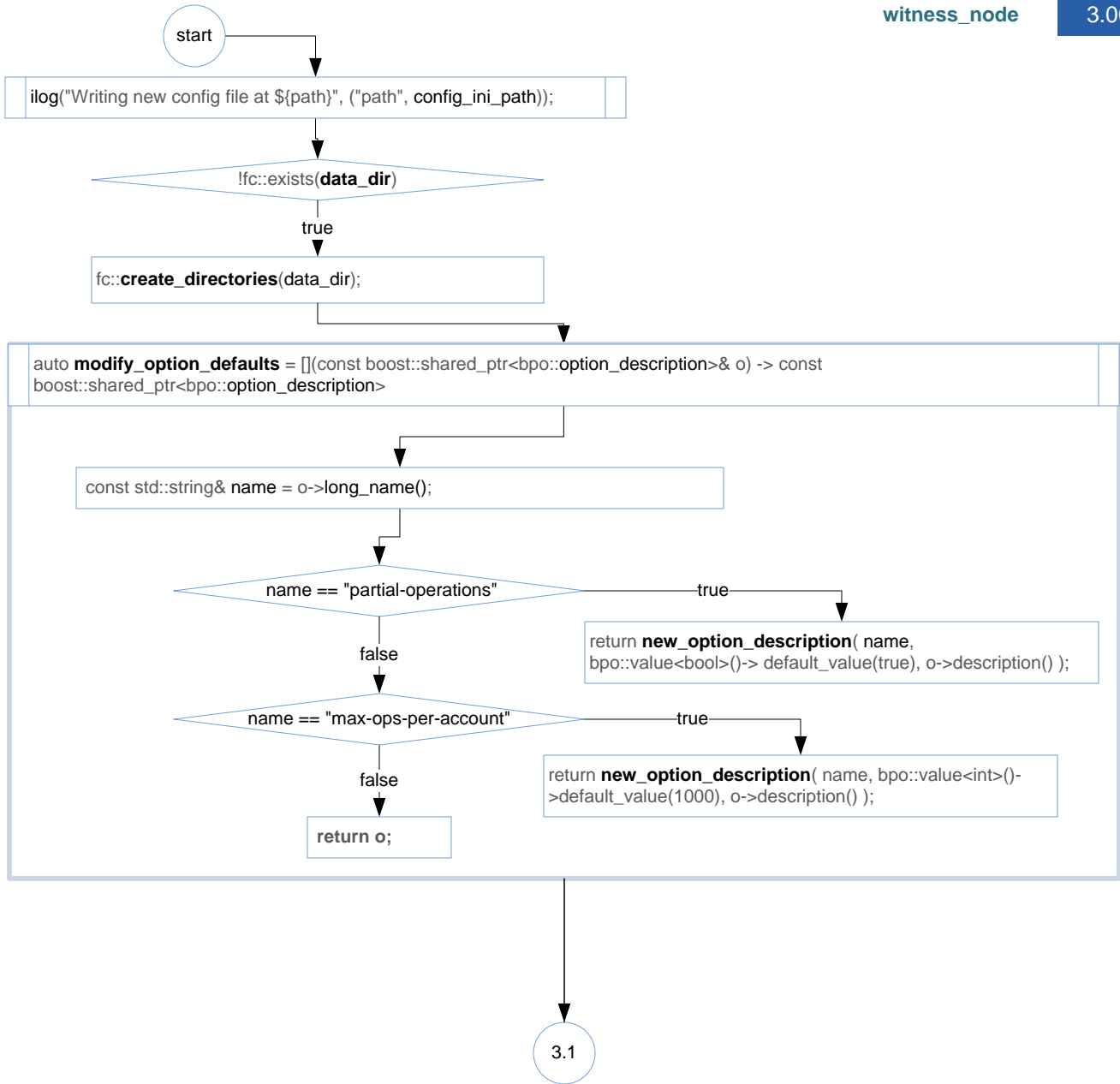
```
static void load_config_file( const fc::path& config_ini_path, const bpo::options_description& cfg_options, bpo::variables_map& options )
```



```
static void create_new_config_file( const fc::path& config_ini_path, const fc::path& data_dir, const bpo::options_description& cfg_options )
```

witness_node

3.00



3.1

```
deduplicator dedup(modify_option_defaults);
std::ofstream out_cfg(config_ini_path.preferred_string());
```

```
const boost::shared_ptr<bpo::option_description> opt : cfg_options.options()
```

true

```
const boost::shared_ptr<bpo::option_description> od = dedup.next(opt);
```

if(!**od**)

true

continue;

false

!od->description().empty()

true

```
out_cfg << "# " << od->description() << "\n";
```

false

```
boost::any store;
```

!od->semantic()->apply_default(**store**)

true

```
out_cfg << "# " << od->long_name() << " = \n";
```

false

```
auto example = od->format_parameter();
```

// This is a boolean switch

example.empty()

true

```
out_cfg << od->long_name() << " = " << "false\n";
```

false

// The string is formatted "arg (=interesting part)"

```
example.erase(0, 6);
example.erase(example.length()-1);
out_cfg << od->long_name() << " = " << example << "\n";
```

```
out_cfg << "\n";
```

false

```
write_default_logging_config_to_stream(out_cfg);
```

```
out_cfg.close();
```

// read the default logging config we just wrote out to the file and start using it

```
fc::optional<fc::logging_config> logging_config = load_logging_config_from_ini_file(config_ini_path);
```

logging_config

true

```
fc::configure_logging(*logging_config);
```

end

int main(int argc, char** argv)

witness_node

1.00

start

```
app::application* node = new app::application()
fc::oexception unhandled_exception;
```

try

```
bpo::options_description app_options("Graphene Witness Node");
bpo::options_description cfg_options("Graphene Witness Node");

app_options.add_options()
("help,h", "Print this help message and exit.")
("data-dir,d", bpo::value<boost::filesystem::path>()->default_value("witness_node_data_dir"),
"Directory containing databases, configuration file, etc.")
("version,v", "Display version information")
;

bpo::variables_map options;

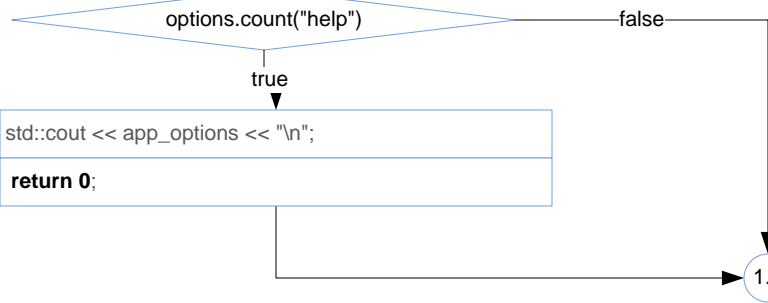
auto witness_plug = node->register_plugin<witness_plugin::witness_plugin>();
auto debug_witness_plug = node->register_plugin<debug_witness_plugin::debug_witness_plugin>();
auto history_plug = node->register_plugin<account_history::account_history_plugin>();
auto elasticsearch_plug = node->register_plugin<elasticsearch::elasticsearch_plugin>();
auto market_history_plug = node->register_plugin<market_history::market_history_plugin>();
auto delayed_plug = node->register_plugin<delayed_node::delayed_node_plugin>();
auto snapshot_plug = node->register_plugin<snapshot_plugin::snapshot_plugin>();
auto es_objects_plug = node->register_plugin<es_objects::es_objects_plugin>();
auto grouped_orders_plug = node->register_plugin<grouped_orders::grouped_orders_plugin>();
```

try

```
bpo::options_description cli, cfg;
node->set_program_options(cli, cfg);
app_options.add(cli);
cfg_options.add(cfg);
bpo::store(bpo::parse_command_line(argc, argv, app_options), options);
```

catch ((const boost::program_options::error& e) const fc::exception&)

```
std::cerr << "Error parsing command line: " << e.what() << "\n";
return 1;
```



int main(int argc, char** argv)

witness_node

1.00

1.1

options.count("version")

true

false

```

std::cout << "Version: " << graphene::utilities::git_revision_description << "\n";
std::cout << "SHA: " << graphene::utilities::git_revision_sha << "\n";
std::cout << "Timestamp: " << fc::get_approximate_relative_time_string(fc::time_point_sec(graphene::utilities::git_revision_unix_timestamp))
<< "\n";
std::cout << "SSL: " << OPENSSSL_VERSION_TEXT << "\n";
std::cout << "Boost: " << boost::replace_all_copy(std::string(BOOST_LIB_VERSION), "_", ".") << "\n";
std::cout << "Websocket++: " << websocketpp::major_version << "." << websocketpp::minor_version << "." << websocketpp::patch_version
<< "\n";

```

return 0;

fc::path data_dir;

options.count("data-dir")

true

false

data_dir = options["data-dir"].as<boost::filesystem::path>();

data_dir.is_relative()

true

false

data_dir = fc::current_path() / data_dir;

fc::path config_ini_path = data_dir / "config.ini";

!fc::exists(config_ini_path)

true

3.00

false

create_new_config_file(config_ini_path, data_dir, cfg_options);

2.00

```

load_config_file( config_ini_path, cfg_options, options );
bpo::notify(options);
node->initialize(data_dir, options);
node->initialize_plugins( options );

node->startup();
node->startup_plugins();

```

fc::promise<int>::ptr exit_promise = new fc::promise<int>("UNIX Signal Handler");

1.2

1.2

```

fc::set_signal_handler([&exit_promise](int signal) {
    elog("Caught SIGINT attempting to exit cleanly");
    exit_promise->set_value(signal);
}, SIGINT);

fc::set_signal_handler([&exit_promise](int signal) {
    elog("Caught SIGTERM attempting to exit cleanly");
    exit_promise->set_value(signal);
}, SIGTERM);

```

```

ilog("Started BitShares node on a chain with ${h} blocks.", ("h", node->chain_database()->head_block_num()));
ilog("Chain ID is ${id}", ("id", node->chain_database()->get_chain_id()));

```

```

int signal = exit_promise->wait();
ilog("Exiting from signal ${n}", ("n", signal));
node->shutdown_plugins();
node->shutdown();

delete node;

return 0;

```

catch(const fc::exception& e)

// deleting the node can yield, so do this outside the exception handler

```
unhandled_exception = e;
```

