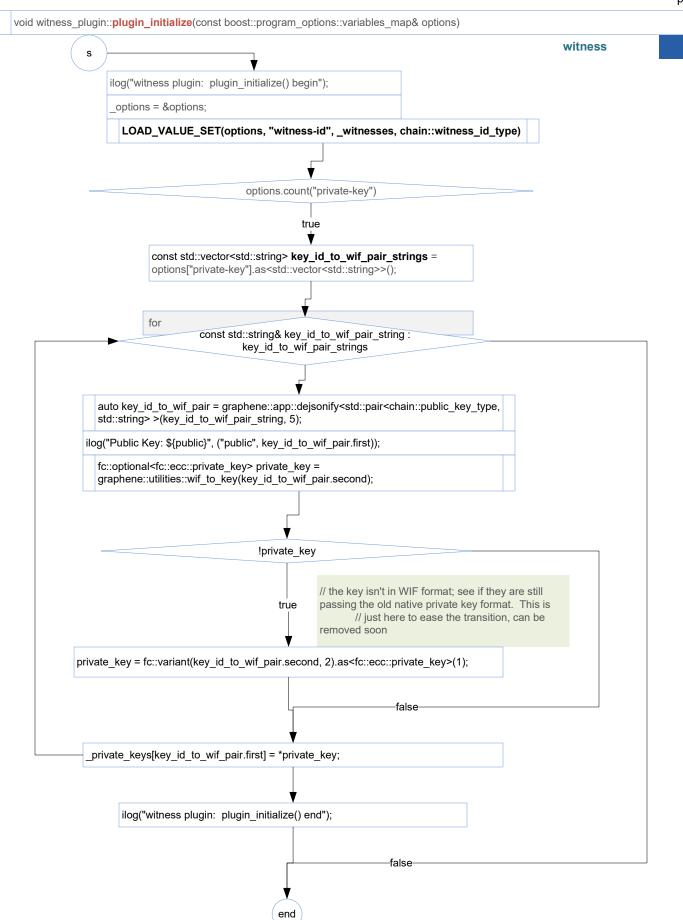
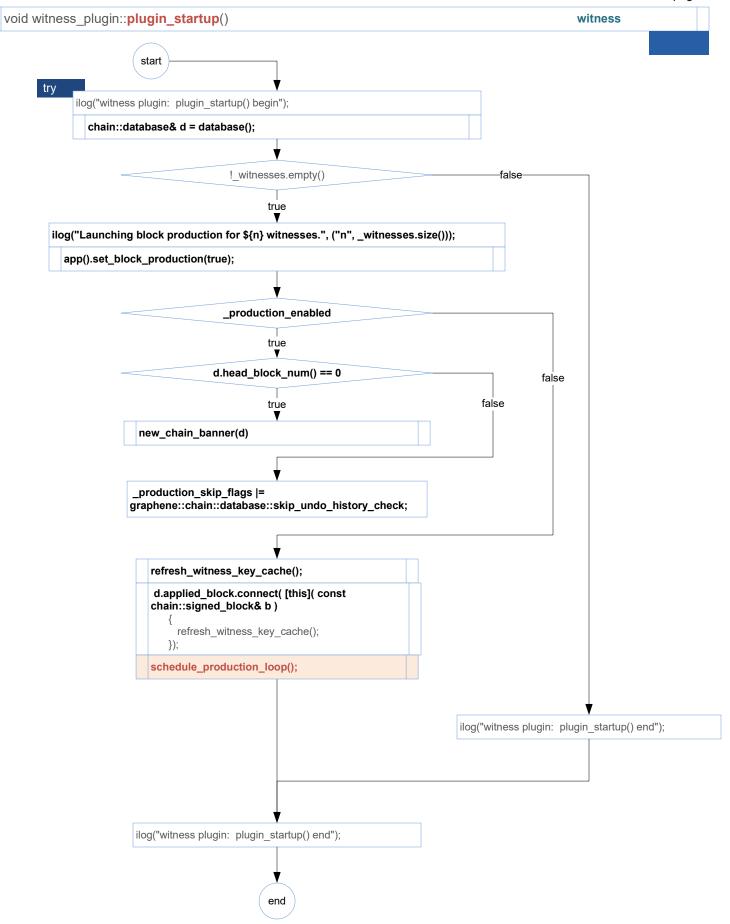


```
page 4
void witness_plugin::plugin_set_program_options(
 boost::program_options::options_description& command_line_options,
 boost::program_options::options_description& config_file_options)
                             start
          auto default_priv_key = fc::ecc::private_key::regenerate(fc::sha256::hash(std::string("nathan")));
       string witness_id_example = fc::json::to_string(chain::witness_id_type(5));
     command line options.add options()
         ("enable-stale-production", bpo::bool_switch() ->notifier([this](bool e){_production_enabled = e;}),
    "Enable block production, even if the chain is stale.") ("required-participation", bpo::bool_switch()
    ->notifier([this](int e){_required_witness_participation = uint32_t(e*GRAPHENE_1_PERCENT);}),
    "Percent of witnesses (0-99) that must be participating in order to produce blocks") ("witness-id,w", bpo::value<vector<string>>()-
    >composing()->multitoken(),
    ("ID of witness controlled by this node (e.g. " + witness_id_example + ", quotes are required, may specify multiple times)").c_str())
         ("private-key", bpo::value<vector<string>>()->composing()->multitoken()
     -> DEFAULT_VALUE_VECTOR(std::make_pair(chain::public_key_type(default_priv_key.get_public_key()),
    graphene::utilities::key_to_wif(default_priv_key))), "Tuple of [PublicKey, WIF private key] (may specify multiple times)")
                          config_file_options.add(command_line_options);
std::string witness_plugin::plugin_name()const
```

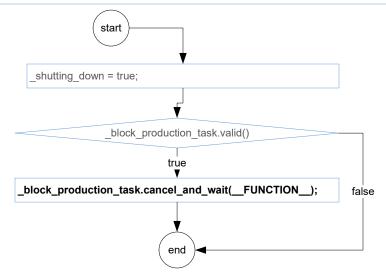
return "witness";



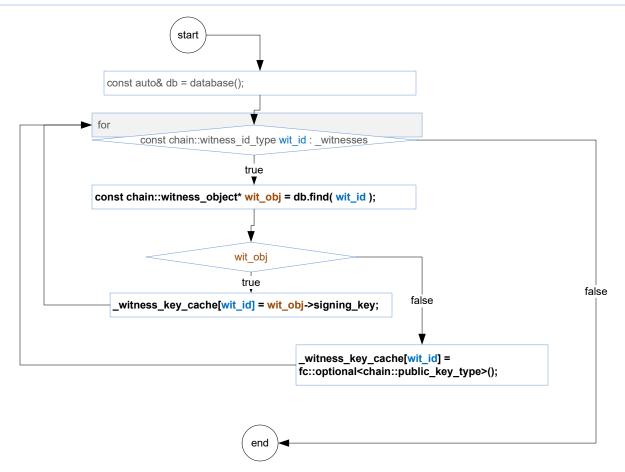


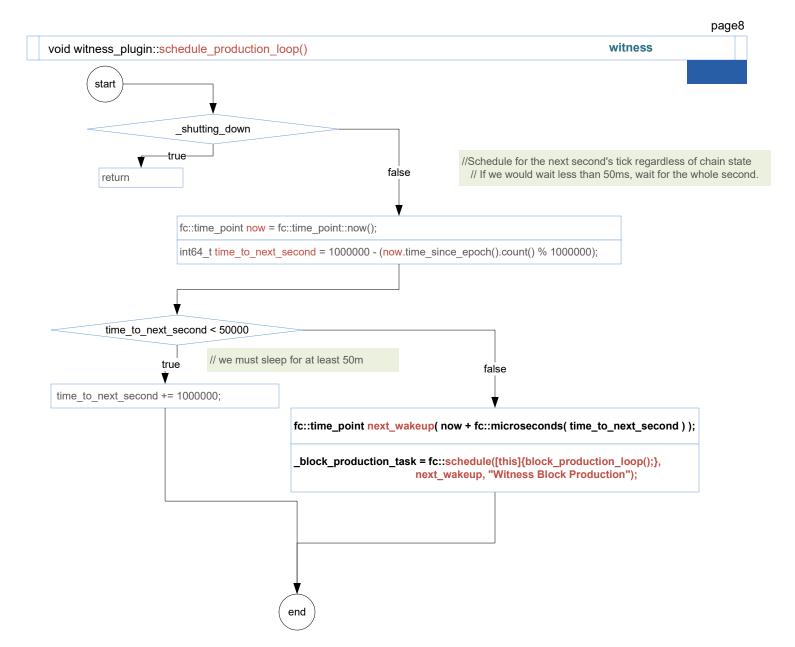
stop_block_production();

witness_plugin::stop_block_production()



void witness_plugin::refresh_witness_key_cache()





```
block production condition::block production condition enum witness plugin::block production loop()
                                                                                                                                      witness
              start
                 block_production_condition::block_production_condition_enum result;
                 fc::limited mutable variant object capture( GRAPHENE MAX NESTED OBJECTS );
                                        _shutting_down
                                                                                                               false
                                     -true
                                                                                            try
                                                                                     result = maybe_produce_block(capture);
 result = block_production_condition::shutdown;
                                                                         catch( const
                                                                      fc::exception& e)
                                                              elog("Got exception while generating block:\n${e}", ("e",
                                                              e.to_detail_string()));
                                                              block_production_condition::exception_producing_block;
switch
                    result
     case block_production_condition::produced:
      ilog("Generated block #${n} with ${x} transaction(s) and timestamp ${t} at time ${c}", (capture));
     case block_production_condition::not_synced:
      ilog("Not producing block because production is disabled until we receive a recent block (see: --enable-stale-production)");
      break:
     case block production condition::not my turn:
      break;
     case block_production_condition::not_time_yet:
      break:
     case block production condition::no private key:
      ilog("Not producing block because I don't have the private key for ${scheduled_key}", (capture) );
      break:
     case block production condition::low participation:
      elog("Not producing block because node appears to be on a minority fork with only ${pct}% witness participation", (capture));
      break:
     case block production condition::lag:
      elog("Not producing block because node didn't wake up within 2500ms of the slot time.");
      break:
     case block_production_condition::exception_producing_block:
      elog( "exception producing block" );
      break;
     case block_production_condition::shutdown:
      ilog( "shutdown producing block" );
      return result;
     default:
      elog( "unknown condition ${result} while producing block", ("result", (unsigned char)result) );
      break;
        schedule_production_loop();
      return result;
```

 $block_production_condition::block_production_condition_enum\ witness_plugin:: \\ \underline{maybe_produce_block}($ fc::limited_mutable_variant_object& capture)

witness

