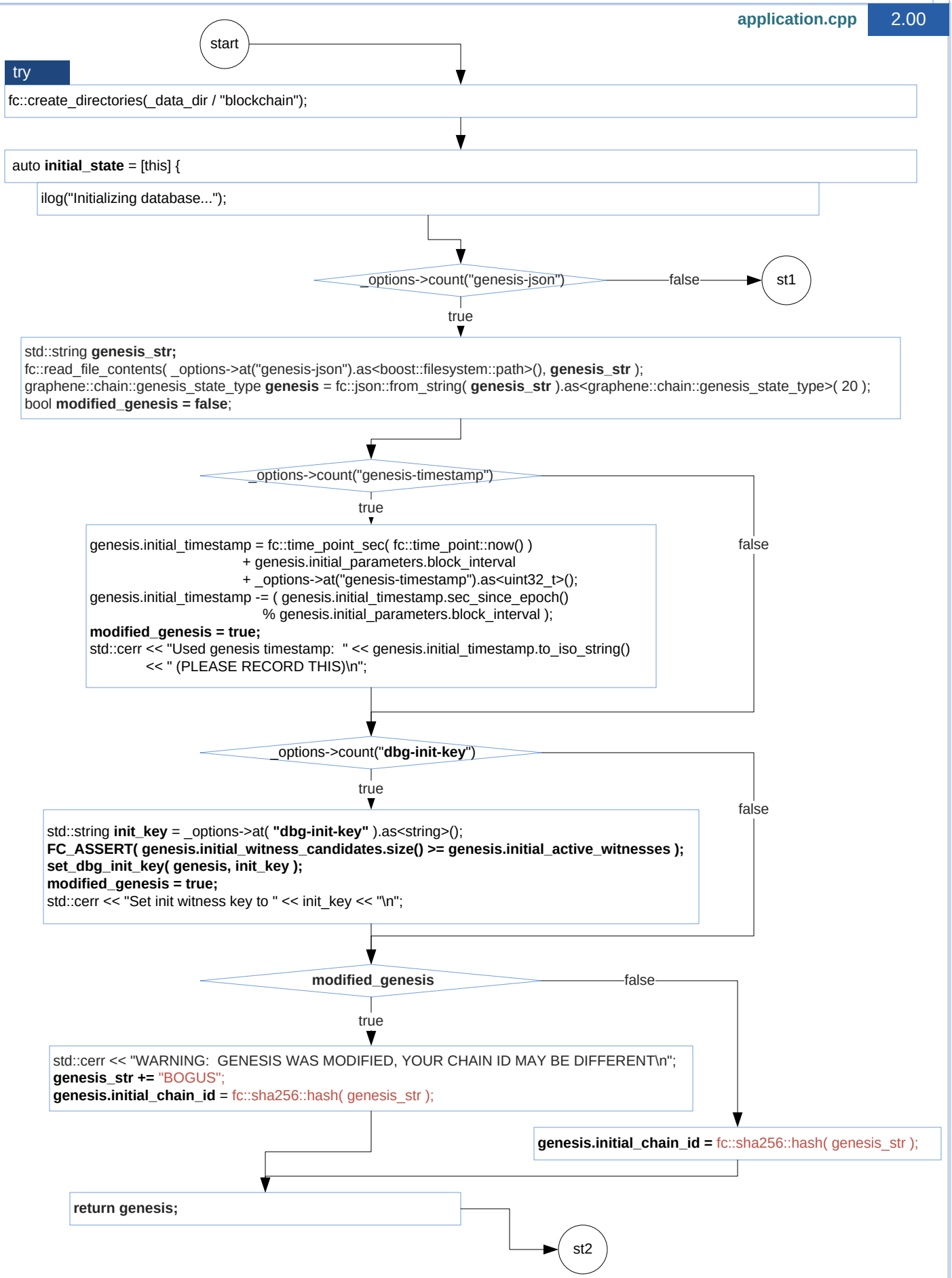
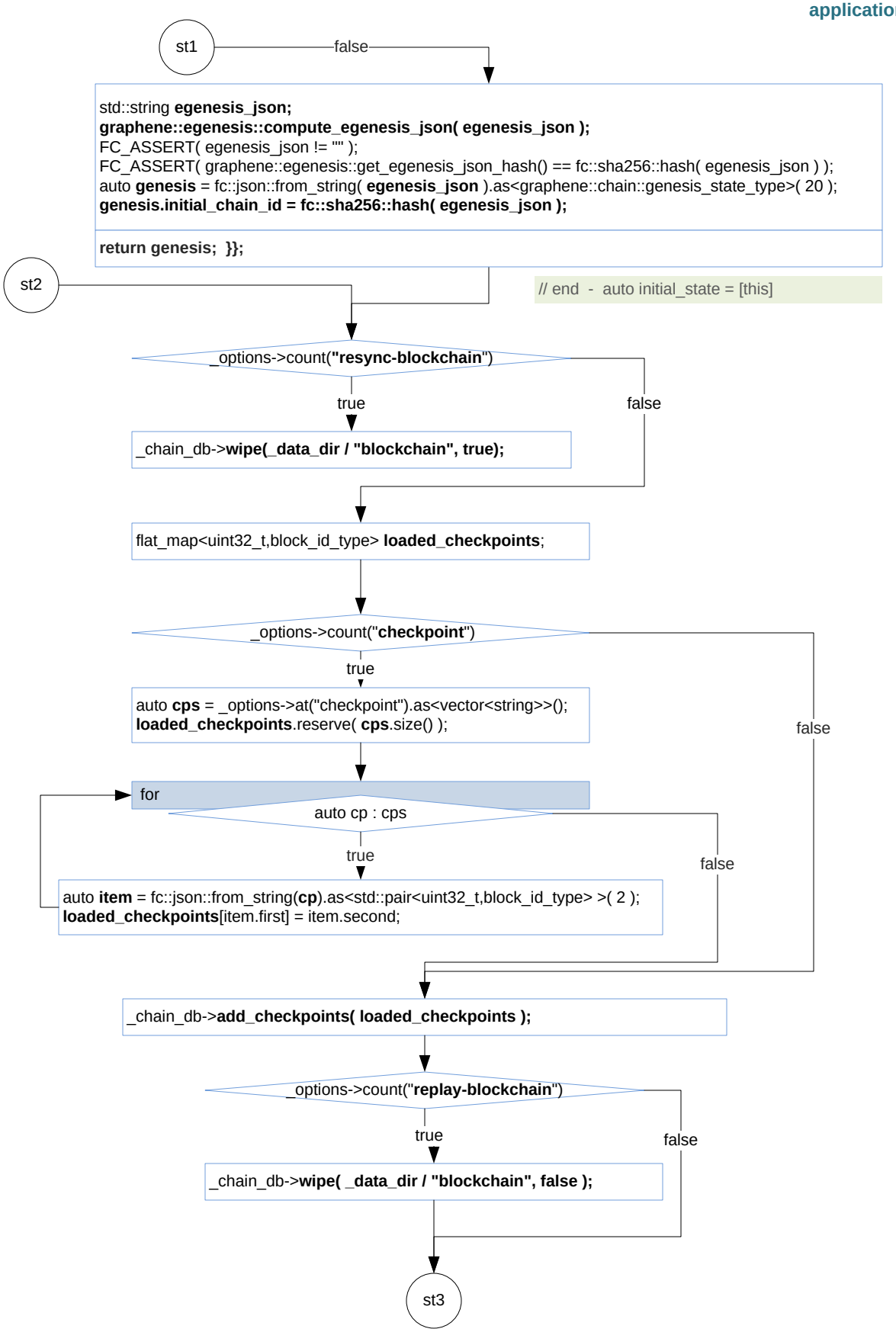
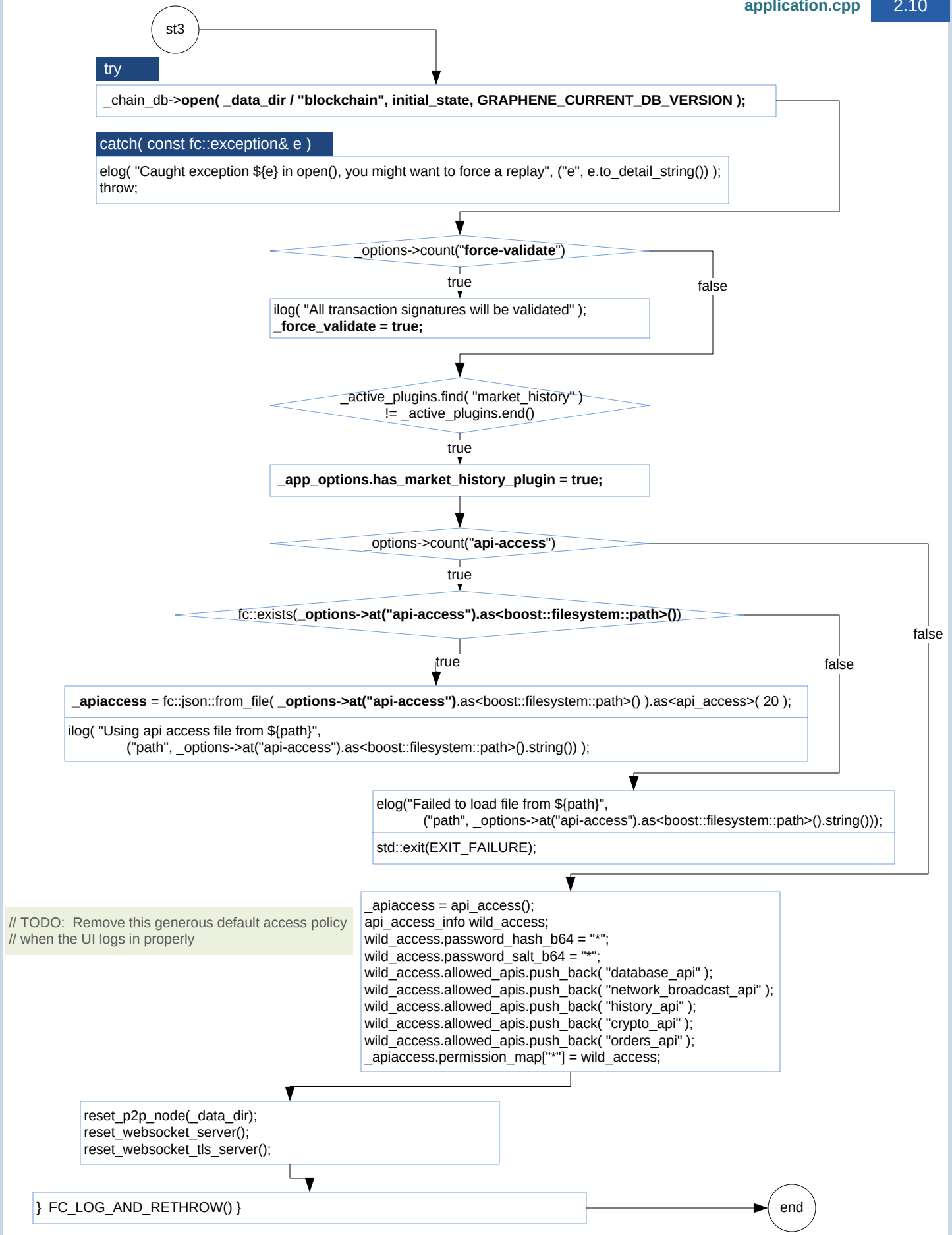
application.cpp 2.00




st1
false
std::string **egenesis_json;**
graphene::egenesis::compute_egenesis_json(egenesis_json);
FC_ASSERT(egenesis_json != "");
FC_ASSERT(graphene::egenesis::get_egenesis_json_hash() == fc::sha256::hash(egenesis_json));
auto **genesis** = fc::json::from_string(**egenesis_json**).as<graphene::chain::genesis_state_type>(20);
genesis.initial_chain_id = fc::sha256::hash(egenesis_json);
return genesis; }};
// end - auto initial_state = [this]
st2
_options->count("**resync-blockchain**")
true
false
_chain_db->**wipe(_data_dir / "blockchain", true);**
flat_map<uint32_t,block_id_type> **loaded_checkpoints**;
_options->count("**checkpoint**")
true
auto **cps** = _options->at("checkpoint").as<vector<string>>();
loaded_checkpoints.reserve(**cps**.size());
for
auto cp : cps
true
auto **item** = fc::json::from_string(**cp**).as<std::pair<uint32_t,block_id_type> >(2);
loaded_checkpoints[item.first] = item.second;
false
false
_chain_db->**add_checkpoints(loaded_checkpoints);**
_options->count("**replay-blockchain**")
true
_chain_db->**wipe(_data_dir / "blockchain", false);**
false
st3


st3
try
_chain_db->open(_data_dir / "blockchain", initial_state, GRAPHENE_CURRENT_DB_VERSION);
catch(const fc::exception& e)
elog("Caught exception ${e} in open(), you might want to force a replay", ("e", e.to_detail_string()));
throw;
_options->count("force-validate")
true
false
ilog("All transaction signatures will be validated");
_force_validate = true;
_active_plugins.find("market_history") != _active_plugins.end()
true
_app_options.has_market_history_plugin = true;
_options->count("api-access")
true
false
fc::exists(_options->at("api-access").as<boost::filesystem::path>())
true
false
_apiaccess = fc::json::from_file(_options->at("api-access").as<boost::filesystem::path>()).as<api_access>(20);
ilog("Using api access file from ${path}", ("path", _options->at("api-access").as<boost::filesystem::path>().string()));
elog("Failed to load file from ${path}", ("path", _options->at("api-access").as<boost::filesystem::path>().string()));
std::exit(EXIT_FAILURE);
// TODO: Remove this generous default access policy
// when the UI logs in properly
_apiaccess = api_access();
api_access_info wild_access;
wild_access.password_hash_b64 = "*";
wild_access.password_salt_b64 = "*";
wild_access.allowed_apis.push_back("database_api");
wild_access.allowed_apis.push_back("network_broadcast_api");
wild_access.allowed_apis.push_back("history_api");
wild_access.allowed_apis.push_back("crypto_api");
wild_access.allowed_apis.push_back("orders_api");
_apiaccess.permission_map["*"] = wild_access;
reset_p2p_node(_data_dir);
reset_websocket_server();
reset_websocket_tls_server();
} FC_LOG_AND_RETHROW() }
end