

cli_wallet - main() Code flow

```
#include <algorithm>
#include <iomanip>
#include <iostream>
#include <iterator>

#include <fc/io/json.hpp>
#include <fc/io/stdio.hpp>
#include <fc/network/http/server.hpp>
#include <fc/network/http/websocket.hpp>
#include <fc/rpc/cli.hpp>
#include <fc/rpc/http_api.hpp>
#include <fc/rpc/websocket_api.hpp>
#include <fc/smart_ref_impl.hpp>

#include <graphene/app/api.hpp>
#include <graphene/chain/config.hpp>
#include <graphene/chain/protocol/protocol.hpp>
#include <graphene/egenesis/egenesis.hpp>
#include <graphene/utilities/key_conversion.hpp>
#include <graphene/wallet/wallet.hpp>

#include <fc/interprocess/signals.hpp>
#include <boost/program_options.hpp>

#include <fc/log/console_appender.hpp>
#include <fc/log/file_appender.hpp>
#include <fc/log/logger.hpp>
#include <fc/log/logger_config.hpp>

#include <graphene/utilities/git_revision.hpp>
#include <boost/version.hpp>
#include <boost/algorithm/string/replace.hpp>
#include <websocketpp/version.hpp>

#ifdef WIN32
# include <signal.h>
#else
# include <csignal>
#endif
```

```
using namespace graphene::app;
using namespace graphene::chain;
using namespace graphene::utilities;
using namespace graphene::wallet;
using namespace std;
namespace bpo = boost::program_options;
```

int main(int argc, char** argv)

cli_wallet

1.00



try

boost::program_options::options_description opts;

```

opts.add_options()
  ("help,h", "Print this help message and exit.")
  ("server-rpc-endpoint,s", bpo::value<string>()->implicit_value("ws://127.0.0.1:8090"), "Server websocket RPC endpoint")
  ("server-rpc-user,u", bpo::value<string>(), "Server Username")
  ("server-rpc-password,p", bpo::value<string>(), "Server Password")
  ("rpc-endpoint,r", bpo::value<string>()->implicit_value("127.0.0.1:8091"), "Endpoint for wallet websocket RPC to listen on")
  ("rpc-tls-endpoint,t", bpo::value<string>()->implicit_value("127.0.0.1:8092"), "Endpoint for wallet websocket TLS RPC to listen on")
  ("rpc-tls-certificate,c", bpo::value<string>()->implicit_value("server.pem"), "PEM certificate for wallet websocket TLS RPC")
  ("rpc-http-endpoint,H", bpo::value<string>()->implicit_value("127.0.0.1:8093"), "Endpoint for wallet HTTP RPC to listen on")
  ("daemon,d", "Run the wallet in daemon mode")
  ("wallet-file,w", bpo::value<string>()->implicit_value("wallet.json"), "wallet to load")
  ("chain-id", bpo::value<string>(), "chain ID to connect to")
  ("version,v", "Display version information");

```

bpo::variables_map options;

bpo::store(bpo::parse_command_line(argc, argv, opts), options);

options.count("help")

true

false

std::cout << opts << "\n";

return 0;

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;

c1

```
int main( int argc, char** argv )
```

cli_wallet

1.00

c1

```

fc::path data_dir;
fc::logging_config cfg;
fc::path log_dir = data_dir / "logs";

fc::file_appender::config ac;
ac.filename      = log_dir / "rpc" / "rpc.log";
ac.flush         = true;
ac.rotate        = true;
ac.rotation_interval = fc::hours( 1 );
ac.rotation_limit  = fc::days( 1 );

std::cout << "Logging RPC to file: " << (data_dir / ac.filename).preferred_string() << "\n";

cfg.appenders.push_back(fc::appender_config( "default", "console", fc::variant(fc::console_appender::config(), 20)));
cfg.appenders.push_back(fc::appender_config( "rpc", "file", fc::variant(ac, 5)));

cfg.loggers = { fc::logger_config("default"), fc::logger_config( "rpc" ) };
cfg.loggers.front().level = fc::log_level::info;
cfg.loggers.front().appenders = {"default"};
cfg.loggers.back().level = fc::log_level::debug;
cfg.loggers.back().appenders = {"rpc"};

idump( (key_to_wif( committee_private_key ) ) );

fc::ecc::private_key nathan_private_key = fc::ecc::private_key::regenerate(fc::sha256::hash(string("nathan")));
public_key_type nathan_pub_key = nathan_private_key.get_public_key();
idump( (nathan_pub_key) );
idump( (key_to_wif( nathan_private_key ) ) )

```

```

wallet_data wdata;

fc::path wallet_file( options.count("wallet-file") ? options.at("wallet-file").as<string>() : "wallet.json");

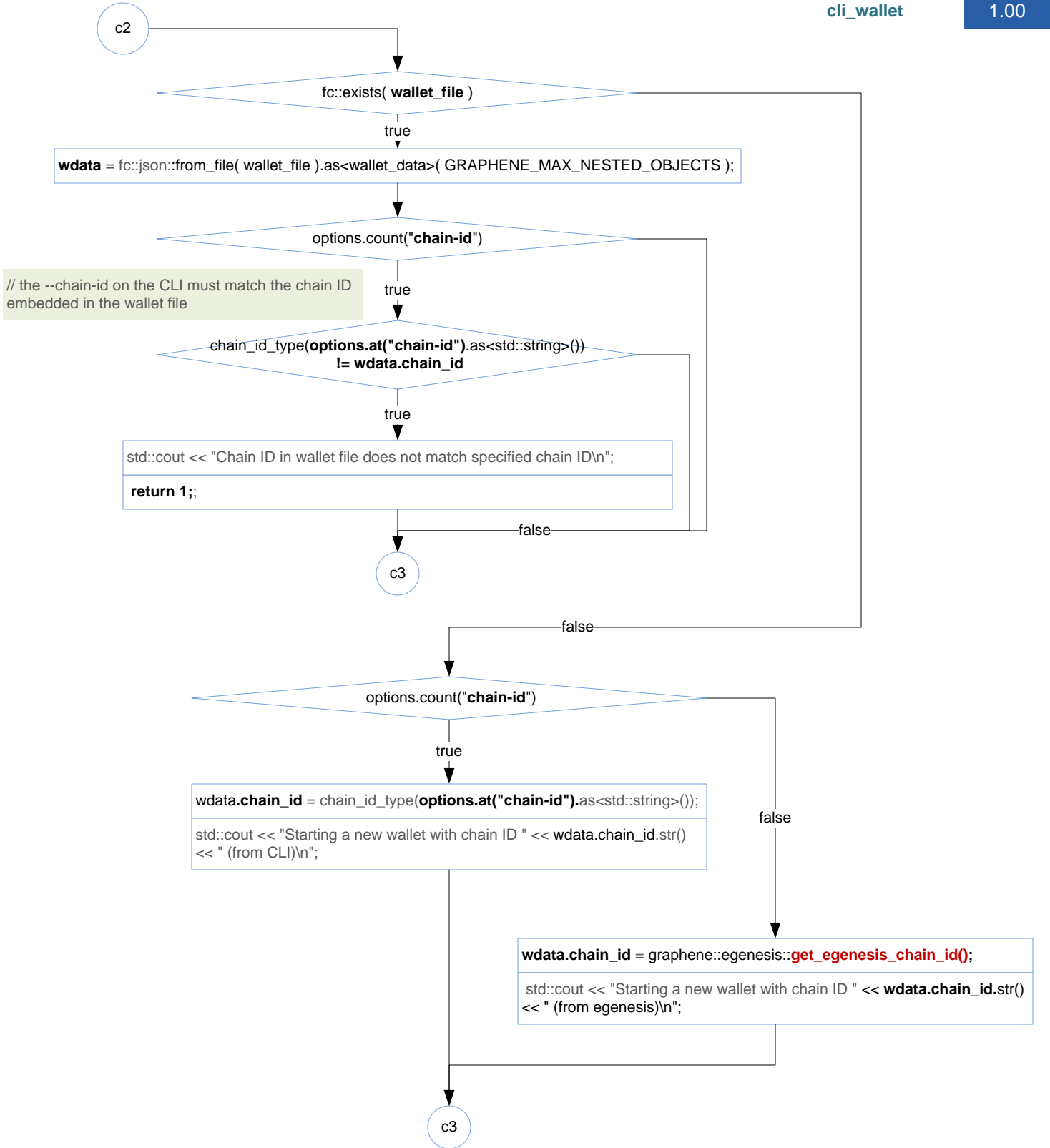
```

c2

```
int main( int argc, char** argv )
```

cli_wallet

1.00



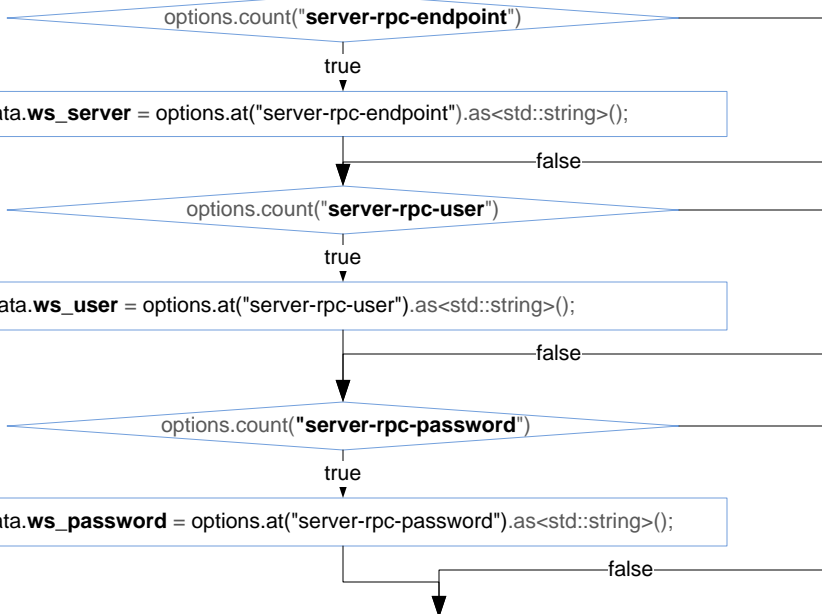
int main(int argc, char** argv)

cli_wallet

1.00

c3

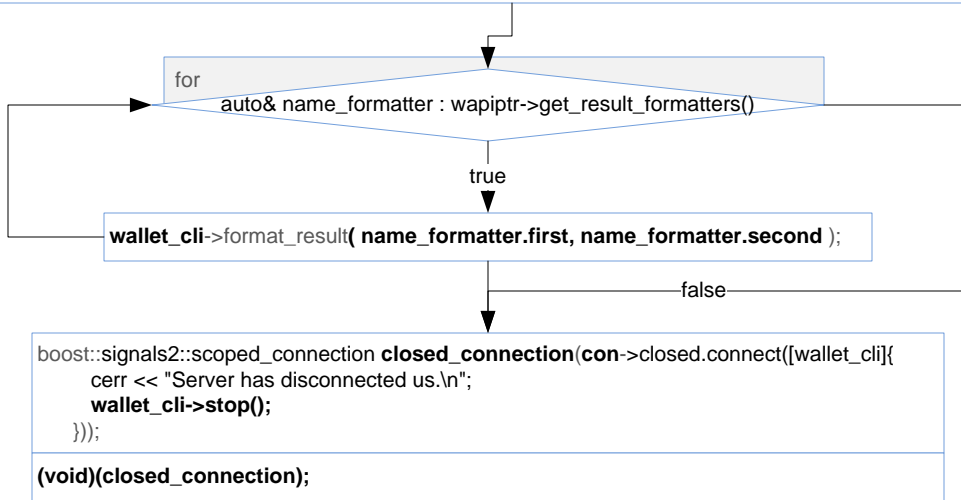
// but allow CLI to override



```

fc::http::websocket_client client;
idump((wdata.ws_server));
auto con = client.connect( wdata.ws_server );
auto apic = std::make_shared<fc::rpc::websocket_api_connection>(*con, GRAPHENE_MAX_NESTED_OBJECTS);
auto remote_api = apic->get_remote_api< login_api >(1);
edump((wdata.ws_user)(wdata.ws_password) );
FC_ASSERT( remote_api->login( wdata.ws_user, wdata.ws_password ), "Failed to log in to API server" );
auto wapiptr = std::make_shared<wallet_api>( wdata, remote_api );
wapiptr->set_wallet_filename( wallet_file.generic_string() );
wapiptr->load_wallet_file();
fc::api<wallet_api> wapi(wapiptr);
auto wallet_cli = std::make_shared<fc::rpc::cli>( GRAPHENE_MAX_NESTED_OBJECTS );

```

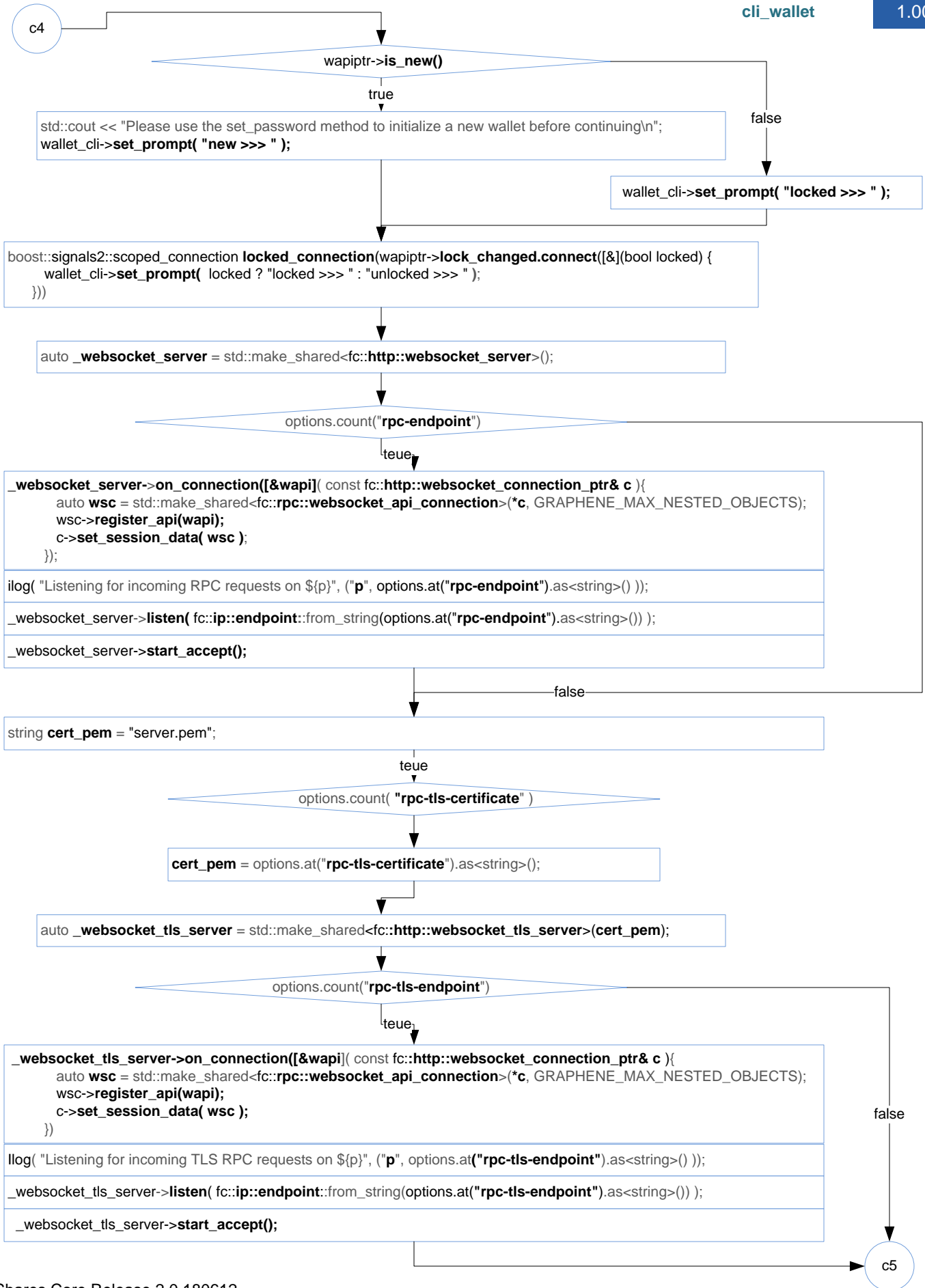


c4

int main(int argc, char** argv)

cli_wallet

1.00



int main(int argc, char** argv)

cli_wallet

1.00

c5

```
auto _http_server = std::make_shared<fc::http::server>();
```

options.count("rpc-http-endpoint")

true

```
ilog( "Listening for incoming HTTP RPC requests on ${p}", ("p", options.at("rpc-http-endpoint").as<string>() ) );
_http_server->listen( fc::ip::endpoint::from_string( options.at( "rpc-http-endpoint" ).as<string>() ) );
```

// due to implementation, on_request() must come AFTER listen()

```
_http_server->on_request(
  [&wapi]( const fc::http::request& req, const fc::http::server::response& resp )
  {
    std::shared_ptr< fc::rpc::http_api_connection > conn =
      std::make_shared< fc::rpc::http_api_connection >( GRAPHENE_MAX_NESTED_OBJECTS );
    conn->register_api( wapi );
    conn->on_request( req, resp );
  } );
```

false

!options.count("daemon")

true

```
wallet_cli->register_api( wapi );
wallet_cli->start();
wallet_cli->wait();
```

false

```
fc::promise<int>::ptr exit_promise = new fc::promise<int>("UNIX Signal Handler");
fc::set_signal_handler([&exit_promise](int signal) {
  exit_promise->set_value(signal);
}, SIGINT);
ilog( "Entering Daemon Mode, ^C to exit" );
exit_promise->wait();
```

```
wapi->save_wallet_file(wallet_file.generic_string());
locked_connection.disconnect();
closed_connection.disconnect();
```

catch (const fc::exception& e)

```
std::cout << e.to_detail_string() << "\n";
return -1;
```

```
return 0;
```

end