Table Of Contents

Introduction

C++ Client is a lightweight gateway to GridGain nodes. Client communicates with grid nodes via REST or TCP binary protocol and provides reduced but powerful subset of GridGain API. C++ Client allows to use GridGain features from devices and environments where fully-functional GridGain node could not (or should not) be started.

The client was tested on following platforms:

- Ubuntu 12.04, compiler: gcc version 4.6.3 (Ubuntu/Linaro 4.6.3-1ubuntu5)
- Mac OS X 10.7.3, compiler: gcc version 4.2.1
- Windows 7, Windows SDK v 7.1

Common Requirements

Third-party Libraries

Google Protobuf

Google protobuf can be obtained from here: http://code.google.com/p/protobuf/downloads/list. Get the latest. Check the included `README.txt` file for build instructions. Make sure you do both: build the binaries and then install them into the default location (usually `/usr/local` on Linux).

When building on Windows using Microsoft Visual Studio 2010 (and possibly other versions), you may run into a compilation error caused by the gtest project (one of the project in the protobuf solution). You can find out more about the issue here:

http://code.google.com/p/googletest/issues/detail?id=217. To work around the issue you should build only the libprotobuf project rather than the entire solution.

Boost

Boost C++ library. On Linux, a stable version can be used (download from http://boost.org). On Windows, Boost.Atomic library is required, which, for the moment, can only be taken from SVN: (do `svn co http://svn.boost.org/svn/boost/trunk boost` to check it out to `boost` directory). The directory with compiled Boost library is called `BOOST_HOME`.

Building Boost on Linux

Change to the Boost installation directory and build the required libraries. The command below will create the debug and release versions of the required libraries in the default location (which is usually `/usr/local` on Linux).

```
cd $BOOST_HOME
./bootstrap.sh
./b2 --with-thread --with-system --with-test link=static,shared install
```

Building Boost on Windows

32-bit libraries:

Launch Windows SDK 7.1 command prompt and change to the Boost home directory. If you're on 64-bit Windows OS you'll need to set the build environment to 32-bit mode as follows:

```
setenv /x86
```

Next, run the following commands:

```
bootstrap.bat
b2 --with-thread --with-atomic --with-system --with-test link=static,shared
--prefix=c:\boost-x86 install
```

This will create the debug and release version of the required libraries in the specified location `c:\boost-x86` on your system. You can use any path prefix you like.

Alternatively, you can download the pre-built Boost libraries from this site: http://boost.teeks99.com

64-bit libraries:

Launch Windows SDK 7.1 command prompt and change to the Boost home directory. Make sure that your current environment as reported by the prompt is x64. If it's not you'll need to set the build environment to 64-bit mode as follows:

```
setenv /x64
```

Next, run the following commands:

```
bootstrap.bat
b2 --with-thread --with-atomic --with-system --with-test link=static,shared
--prefix=c:\boost-x64 address-model=64 install
```

This will create the debug and release version of the required libraries in the specified location `c:\boost-x64` on your system. You can use any path prefix you like.

Alternatively, you can download the pre-built Boost libraries from this site: http://boost.teeks99.com

Building the Client Library

Building on Windows

Building on Windows with MS Visual C++ 2010.

IMPORTANT You must link against a protobuf library with a matching build configuration. Specifically, the Debug build of the GridGain client must be linked against the Debug build of the protobuf library. Similarly - for the Release build. Failure to do so will result in the linker errors.

In order to be able to compile you need to have OpenSSL installed. You can download a binary distribution from here: http://www.openssl.org/related/binaries.html. Also, in order to be able to generate doxygen documentation you'll need a Win32 doxygen installed.

By default, the client library links against Boost's static libraries. In order to build against Boost's shared libraries, specify the `BOOST_ALL_DYN_LINK` preprocessor directive and rebuild the client.

You should have MS Visual C++ 2010 available on your system. A copy of Microsoft Visual C++ 2010 Express can be downloaded from the Microsoft web site free of charge.

Open Visual Studio Command prompt and change to the vsproject/ directory under the GridGain client installation directory.

You now need to define the following environment variables:

1. `BOOST_HOME` - should point to the top level directory of the Boost library.
2. `PROTOBUF_DEBUG_HOME` - should point to the location where the Debug build of the protobuf package is installed. This variable is used by the Debug builds of GridGain C++ client.
3. `PROTOBUF_RELEASE_HOME` - should point to the location where the Release build of the protobuf package is installed. This variable is used by the Release builds of GridGain C++

client.
4. `OPENSSL_HOME` - installation directory of OpenSSL binary.

With regard to the protobuf library, the GridGain client library's MS VC++ project expects the protobuf headers to be located under `PROTOBUF_(RELEASE|DEBUG)_HOME/include` and the `libprotobuf.lib` - under `PROTOBUF_(RELEASE|DEBUG)_HOME/lib`.

You can now build the client's debug version using the following command:

```
msbuild gridgain-client.sln /p:Configuration=Debug /p:Platform="Win32"
```

In order to build the release version, replace "Debug" with "Release" in the above command. In order to build a 64-bit library, replace "Win32" with "x64" in the above command.

It's also possible to build the client binary from the Visual C++ IDE. Start the IDE and open the `vsproject/gridgain-client.sln` solution and press F7 to build it.

Building on Linux

For builing on Linux the client uses the GNU Autotools toolchain. Go to the `main/` directory and run:

```
./configure
```

You can also do

```
./configure --help
```

to see all available command line options. Once the configure script finishes, you can run a make:

```
make install
```

This will build and install the client library and the headers into the default location on your system (which is usually `/usr/local`).

Doxygen documentation

You can generate the documentation by going into the `main/` directory and doing:

```
doxygen doxygen.conf
```

This will produce the package documentation under docs/html directory. To view the

documentation open `docs/html/index.html` in your favorite browser.

On Linux you can also generate the documentation by doing this:

```
make doxygen-doc
```

You must be in the `main/` directory.

Running

Configuring log

You can control the client's log level by setting the `GRIDGAIN_CPP_CLIENT_LOG_LEVEL` environment variable to one of the following values:

1. (error),
2. (warning),
3. (info), and
4. (debug).

By default, the log level is 3. The client sends all log messages to stdout.