QpidCppWindowsBuild

These notes were contributed by Vince Seavello and include details of getting a build from the svn repository to work on:

Windows:

Windows Vista (64 bit)
Windows Server 2008 (64 bit).
Windows XP (32 bit)

Visual Studio:

Visual Studio 2008 Express (32 bit C++ package)
Visual Studio Team Suite 2008 (32 bit and 64 bit C++ packages)
Visual Studio 2008 Professional (32 bit and 64 bit C++ packages)

Note: Step 2.5.1 (nmake /f protocol_gen.mak) is not needed for the M4-release build. It's only needed when checking sources out of svn.

1) Introduction

The following are notes for building and testing QPID M4 on Windows.

This document is split into 4 sections:

- setup for building QPID M4 on Windows. This includes obtaining sources, supporting technologies, installation and configuration of the build environment.
- building QPID M4 on Windows.
- running the Linux test suites against the Windows QPID broker.
- building the tests on a Windows system.

These notes are intended to highlight issues discovered during the build and testing of QPID on Windows. It's a work in progress.

Comments welcome.

2) Setup of QPID M4 on Windows

2.1) Obtaining the sources

M4 C++ broker and client source archive is available on the QPID web site.

http://www.apache.org/dist/qpid/M4/qpid-cpp-M4.tar.gz

The sources contained here will build a Static_Debug version of the broker and client libraries. To be able to build a Static_Release version, replace the file cpp/src/qpid/InlineAllocator.h with the M5 version of the file:

https://svn.apache.org/repos/asf/qpid/trunk/qpid/cpp/src/qpid/InlineAllocator.h

2.2) Setting up for the build

Builds have been taking place using a variety of platforms and versions of Visual Studio. Platforms include:

Windows:

Windows Vista (64 bit)
Windows Server 2008 (64 bit).
Windows XP (32 bit)

Visual Studio:

Visual Studio 2008 Express (32 bit C++ package)
Visual Studio Team Suite 2008 (32 bit and 64 bit C++ packages)
Visual Studio 2008 Professional (32 bit and 64 bit C++ packages)

2.3) Technologies Dependencies

2.3.1) boost

The stated requirement is boost 1.35.0. 1.36.0 is also being tested in our labs. There are only a few version compatibility issues detected.

You can find install images for boost at:

http://www.boostpro.com/boost_1_35_0_setup.exe http://www.boostpro.com/boost_1_36_0_setup.exe

Once you've installed one of these, set your environment variables:

for 1.35.0:
BOOST_ROOT = C:\Program Files (x86)\boost\boost_1_35_0
BOOST_VERSION = 103500

for 1.36.0:
BOOST_ROOT = C:\Program Files (x86)\boost\boost_1_36_0
BOOST_VERSION = 103600

Note: "Program Files (x86)" is "Program Files" on the 32 bit system.
The boost libs are 32 bit. When building on a 64 bit system the 32
bit boost libs are installed in "Program Files (x86)". 64 bit boost
build is being looked into.

2.3.2) python

Use Python 2.6, which is the stated requirement for the QPID build.
Pick up a copy at:

http://www.python.org/ftp/python/2.6/python-2.6.msi

You must include the python directory in the environment variable PATH.
Typically, python is installed in:

C:\python26

test to see that python is installed correctly and that you have the
correct version:

C:\>python --version
Python 2.6.1

2.3.3) ruby

Use Ruby 1.8.6, which is the required version. Pick up a copy at:

http://rubyforge.org/frs/?group_id=167

You must include the ruby directory in the environment variable PATH.
Typically, ruby is installed in:

C:\ruby\bin

test to see that ruby is installed correctly and that you have the
correct version:

C:\>ruby -v
ruby 1.8.6 (2007-09-24 patchlevel 111) i386-mswin32

2.4) Build tools

The QPID build environment on Windows are centered around Visual Studio C++
Development tools. There is an effort to reduce the overhead of the QPID
build so developers who are unfamiliar with the Visual Studio IDE can build
and test on Windows with as minimal an effort as possible.

2.4.1) Visual Studio

Microsoft makes Visual Studio Express available at no charge. You can find
it online at the following location:

http://www.microsoft.com/downloads/details.aspx?FamilyID=3254c868-bcb9-412c-95c6-d100c872ec60&DisplayLang=en

2.5) M4 preparation

2.5.1) Initial environment configuration

Most of the prep is in the installation of the helper apps, Visual Studio,
source extraction and patching. When that's all done, there is one more step
you still need to do before you can build anything.

Using the Visual Studio's Command Prompt, change directory to qpid/cpp/src and run:

nmake protocol_gen.mak

2.5.2) source changes

The described source fix needed to be able to get a clean Static_Release build from M4 is listed above. This fix wasn't available before M4 was finalized, so it was added in M5.

Other fixes may be available in M5 to address other issues, but haven't been tried in this environment.

Any changes made to source files during this investigation haven't been submitted. They are presented here for what they're worth. Diffs of source changes are appended at the end of this document.

2.5.3) Debug vs Release

When building QPID, you may want to change from building static debug to static release. There are several ways to do this. Look in the tool bar at the top of the Visual Studio window. Locate the pulldown selection box that contains the value "Debug" or "Release". Change the value to whichever active solution configuration you would like to produce. All built objects/executables will be found in the subdirectories Static_Debug or Static_Release, depending on your active solution configuration.

3) Building QPID M4 on Windows

3.1) QPID Project Files

Visual Studio uses special configuration files to identify the "solution" and "project" parameters. These are made available in .sln and .vcproj files. There are a smattering of these files in the M4 source tree, and more are being added to the M5 tree as time goes on.

For QPID, look for the following:

qpid/cpp/src/qpid.sln
qpid/cpp/src/broker.vcproj
qpid/cpp/src/client.vcproj
qpid/cpp/src/common.vcproj
qpid/cpp/src/MaxMethodBodySize.vcproj

When you open the solution, you can choose to build the entire solution, or just a project within. The executables and objects will be in Static_Release or Static_Debug, depending on your build settings.

Select the solution in the solution explorer pane. Right mouse click and choose "Build Solution". When complete, you should have built the libraries and qpidbroker.exe.

3.2) Libraries

The Windows QPID M4 build will produce 2 libraries:

qpidclients.lib
qpidcommons.lib

The Linux build will also produce a broker library. The Windows build links the broker source objects directly into the broker, but doesn't add them to a library. A broker library will be needed for the building of some tests.

The build of the broker will have compiled all the source objects which are to live in the broker library. By creating a library by hand we will be able to link the tests that require the broker library. See libs.mk later in this document (section 5.4.1).

Libraries are placed in cpp/src.

Our efforts don't include building the qmfconsole library yet, so no status on that.

4) Testing AMQP from a Linux system

There isn't a "push button" build and test for Windows. In fact, in M4 there are relatively few tests that build for Windows. However, the tests do run on Linux. The Windows broker can be tested by running the test suites from a Linux platform.

4.1) localhost vs distributed

The test suites as they are built and run on Linux assume brokers and tests are running on the localhost. There are a few that try to use ssh to start and stop processes on a remote host. But, the majority of the tests start brokers locally, look for PID files and run the tests against the local brokers.

In order to use these tests against a remote Windows broker, some modifications will be necessary. These changes are outlined later in this document.

4.2) test harnesses

QPID makes use of the boost test framework for some of it's tests, but there are a number of different tools and programming methods used. These are:

make - Building and running the tests from the makefile. Provides a convenient way to produce a push button QPID. The makefiles allow selection of subsets of tests and provide some command parameters to the test framework.

boost - Boost provides libraries and header files for putting together test cases, test suites and overall test execution control.

/bin/sh shell - the "bridge" between make and the test cases is handled through shell scripts and other tools, like valgrind and libtool. There are makefile variables that can control how some of these tools are utilized.

python - There is a separate python directory with test cases and test suites. These tests are initiated from the makefile by calling a wrapper boost test case called python_tests.

C++ - The .cpp source files in the tests directory are the main consumers of the boost framework. They typically provide a common command line interface that every test shares, as well as additional options specific to the test itself. Many of these tests have multiple test cases within and are controlled, to some fashion, by the command line options.

At the top level is the make file. To initiate the tests, type the following:

make check

This will build all the QPID sources and tests. After all has been built, the test suites are run. There are several options you can use when using make to narrow down the running of the tests.

You can select a specific list of test suites by setting the variable TESTS:

make check TESTS=perftest

These command line options have been helpful in isolating the various test suites.

The makefile uses the script run_test to initiate the indicated test. In most cases, the test being requested is front-ended by a shell script generated during the make process.

The run_test script can pass command line options through to the called program, but the makefile isn't set up to send much in the way of options. Many of the lower level test programs have an option that accepts a broker address on the command line. In some cases, the intermediate scripts have a broker option, too.

In order to use the Linux tests against the Windows broker, the tests have to be redirected to the remote broker address. Rather than modify the structure of the Makefile.am file used to generate the final makefile, the broker address can be passed to the tests in the execution environment. The modified command line invocation then looks like this:

BROKER="IP_Address" make check TESTS="fanout_perftest"

The run_test script can be modified to look for a broker address environment variable. Because run_test is used to start many of the test programs, sometimes the broker address should not to be passed through. In particular, the broker address is not necessary when starting and stopping localhost brokers. The run_test script determines when to pass the broker address by looking at the program name.

Alternatively, tests can be run directly. The fanout perftest example can be run like this:

```
./perftest --summary --count 30000 --broker IP_Address \
--mode fanout --npubs 16 --nsubs 16 --size 64
```

The python tests can be run by changing directory to the python test dir and using one of the following command lines:

```
./run-tests --skip-self-test -v -s 0-10-errata \
-I cpp_failing_0-10.txt \
-b 10.197.62.244:5672
./run-tests -v -s 0-10-errata -I cpp_failing_0-10.txt \
-b 10.197.62.244:5672
./run-tests -v -s 0-10-errata -b 10.197.62.244:5672
```

4.3) Changes to Linux tests

Some of the tests, or scripts that initiate the tests, have the localhost address hardcoded, or look for a broker PID in a local file. A majority of the changes made to allow the tests to be run from a Linux system to a Wondows broker deal with this limitation.

The tests that were run successfully against the Windows broker from a Linux system are:

client_test
quick_perftest
quick_topictest
python_tests
run_federation_tests
fanout_perftest
shared_perftest
multiq_perftest
topic_perftest
txtest
latencytest
echotest
benchmark

5) Building tests on Windows

The Linux test environment makes use of tools, scripts and methods that don't work on a Windows platform. These issues will have to be addressed before a "push button" test can be accomplished similar to that on the Linux system.

In the mean time, the test programs can be built on a Windows platform and run by hand. The following sections talk about this.

5.1) quick and dirty modifications

The modifications described here do not represent "best practice" coding methodologies. They are presented for information sake.

The following files required some sort of modification to build on Windows. A diff listing will be added to the end of this document.

cpp/src/tests/ClientSessionTest.cpp
cpp/src/tests/FieldTable.cpp
cpp/src/tests/MessageBuilderTest.cpp
cpp/src/tests/MessageTest.cpp
cpp/src/tests/QueueOptionsTest.cpp
cpp/src/tests/TimerTest.cpp
cpp/src/tests/Uuid.cpp
cpp/src/tests/logging.cpp
cpp/src/tests/perftest.cpp
cpp/src/tests/topic_listener.cpp
cpp/src/tests/topic_publisher.cpp
cpp/src/tests/unit_test.h

While the following needed modification, the have been excluded
from the build due to larger porting issues.

cpp/src/tests/ForkedBroker.h
cpp/src/tests/failover_soak.cpp
cpp/src/tests/ConsoleTest.cpp

5.2) nmake environment

A small handful of make files have been created to use on a Windows
platform to build the various test programs. The nmake utility is used
with these files to perform the compilation.

tests/libs.mk
tests/tests.mk

5.3) diff listings

```
==========================================

cpp/src/tests/ClientSessionTest.cpp
38a39,44
> #if defined(_WIN32)
> # include <windows.h>
> # include <winbase.h>
> #endif
>
>
225a232,234
> #if defined(_WIN32)
> Sleep(1000);
> #else
226a236
> #endif
277a288,290
> #if defined(_WIN32)
> Sleep(2000);
> #else
278a292
> #endif
291a306,308
> #if defined(_WIN32)
> ::Sleep(300* 1);
> #else
292a310
> #endif

==========================================

cpp/src/tests/FieldTable.cpp
25c25,27
< #include <alloca.h>
—
> #if !defined (_WIN32)
> # include <alloca.h>
> #endif

==========================================

cpp/src/tests/MessageBuilderTest.cpp
39c39
< uint64_t id;
—
> boost::uint64_t id;
215c215
< BOOST_CHECK_EQUAL((uint64_t) 1, builder.getMessage()->getPersistenceId());
—
> BOOST_CHECK_EQUAL((boost::uint64_t) 1, builder.getMessage()->getPersistenceId());

==========================================
```

```
cpp/src/tests/MessageTest.cpp
31c31,33
< #include <alloca.h>
—
> #if !defined (_WIN32)
> # include <alloca.h>
> #endif
81,82c83,84
< BOOST_CHECK_EQUAL((uint64_t) data1.size() + data2.size(), msg->contentSize());
< BOOST_CHECK_EQUAL((uint64_t) data1.size() + data2.size(), msg->getProperties<MessageProperties>()->getContentLength());
—
> BOOST_CHECK_EQUAL((boost::uint64_t) data1.size() + data2.size(), msg->contentSize());
> BOOST_CHECK_EQUAL((boost::uint64_t) data1.size() + data2.size(), msg->getProperties<MessageProperties>()->getContentLength());
85c87
< BOOST_CHECK_EQUAL((uint8_t) PERSISTENT, msg->getProperties<DeliveryProperties>()->getDeliveryMode());
—
> BOOST_CHECK_EQUAL((boost::uint8_t) PERSISTENT, msg->getProperties<DeliveryProperties>()->getDeliveryMode());

==========

cpp/src/tests/QueueOptionsTest.cpp
24c24,26
< #include <alloca.h>
—
> #if !defined(_WIN32)
> # include <alloca.h>
> #endif

==========

cpp/src/tests/TimerTest.cpp
78c78
< uint64_t difference = abs(expected - actual);
—
> uint64_t difference = abs((long)(expected - actual));

==========

cpp/src/tests/Uuid.cpp
25c25,27
< #include <alloca.h>
—
> #if !defined(_WIN32)
> # include <alloca.h>
> #endif

==========

cpp/src/tests/logging.cpp
25a26
> # include "qpid/log/windows/SinkOptions.h"
272a274,276
> #if defined(_WIN32)
> qpid::log::windows::SinkOptions sinks("test");
> #else
273a278
> #endif
288a294,296
> #if defined(_WIN32)
> qpid::log::windows::SinkOptions sinks("test");
> #else
289a298
> #endif
347a357,360
> #if defined(_WIN32)
> qpid::log::windows::SinkOptions *sinks =
> dynamic_cast<qpid::log::windows::SinkOptions *>(opts.sinkOptions.get());
> #else
349a363
> #endif

==========
```

```
cpp/src/tests/perftest.cpp
41c41,43
< #include <unistd.h>
—
> #if !defined(_WIN32)
> # include <unistd.h>
> #endif

==========

cpp/src/tests/topic_listener.cpp
43a44,46
> #if defined(_WIN32)
> # include <process.h>
> #endif

==========

cpp/src/tests/topic_publisher.cpp
43c43,45
< #include <unistd.h>
—
> #if !defined(_WIN32)
> # include <unistd.h>
> #endif

==========

cpp/src/tests/unit_test.h
56c56
< #if (BOOST_VERSION < 103600)
—
> #if (BOOST_VERSION <= 103600)
```

5.4) make files

Once the Visual Studio build has been run, the objects for the
broker are available and can be combined into a broker library.
This nmake file can be used in the tests directory to create the
library. This will be necessary to build the unit_test test suite.

From the tests directory you can run nmake with the tests.mk file
listed below to build the tests. This build file does not build
the files:

ConsoleTest.cpp
failover_soak.cpp

5.4.1) lib.mk

```
C89_COMPILER=C:/Program Files (x86)/Microsoft Visual Studio 9.0/VC/bin/cl.exe
C89_LINKER=C:/Program Files (x86)/Microsoft Visual Studio 9.0/VC/bin/link.exe
LINK="link.exe"

OBJEXT=.obj

INCLUDES=/I "C:/Program Files (x86)/boost/boost_1_36_0/include/103600" /I "C:/Program Files (x86)/boost/boost_1_36_0/." /I "." /I ".." /I "../gen"
FLAGS=/D "NDEBUG" /D "WIN32" /D "_CONSOLE" /D "_CRT_NONSTDC_NO_WARNINGS" /D "NOMINMAX" /D "WIN32_LEAN_AND_MEAN" /D
"_SCL_SECURE_NO_WARNINGS" /D "_CRT_SECURE_NO_WARNINGS" /FD /EHsc /MT /W3 /c /TP /wd4244 /wd4800 /wd4290 /wd4355

LIBS=ws2_32.lib secur32.lib rpcrt4.lib kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib
odbc32.lib odbccp32.lib
LDFLAGS=/INCREMENTAL:NO /LIBPATH:"." /LIBPATH:"C:\Program Files (x86)\boost\boost_1_36_0\lib" /DEBUG /SUBSYSTEM:CONSOLE /OPT:REF
/OPT:ICF /DYNAMICBASE /NXCOMPAT /MACHINE:X86 /nologo /errorReport:prompt

RELDIR=Static_Release
ROOTDIR=C:\Users\v-vinsea\Dev\AMQP\qpid-M4\cpp\src\tests

check_PROGRAMS=
check_LTLIBRARIES=
TESTS=
EXTRA_DIST=
CLEANFILES=

.cpp$(OBJEXT):
echo $(CC) /O2 $(INCLUDES) $(FLAGS) /Fo"..\Static_Release\qmfconsole/I386//" /Fd"..\Static_Release\qmfconsole/I386/vc90.pdb" $<

broker_OBJECTS=../Static_Release/broker/I386/Acl.obj \
../Static_Release/broker/I386/Agent.obj \
```

../Static_Release/broker/I386/Binding.obj \
../Static_Release/broker/I386/Bridge.obj \
../Static_Release/broker/I386/Bridge2.obj \
../Static_Release/broker/I386/Broker.obj \
../Static_Release/broker/I386/Broker2.obj \
../Static_Release/broker/I386/BrokerDefaults.obj \
../Static_Release/broker/I386/BrokerSingleton.obj \
../Static_Release/broker/I386/Connection.obj \
../Static_Release/broker/I386/Connection2.obj \
../Static_Release/broker/I386/Connection3.obj \
../Static_Release/broker/I386/ConnectionFactory.obj \
../Static_Release/broker/I386/ConnectionHandler.obj \
../Static_Release/broker/I386/DeliverableMessage.obj \
../Static_Release/broker/I386/DeliveryRecord.obj \
../Static_Release/broker/I386/DirectExchange.obj \
../Static_Release/broker/I386/DtxAck.obj \
../Static_Release/broker/I386/DtxBuffer.obj \
../Static_Release/broker/I386/DtxManager.obj \
../Static_Release/broker/I386/DtxTimeout.obj \
../Static_Release/broker/I386/DtxWorkRecord.obj \
../Static_Release/broker/I386/EventAllow.obj \
../Static_Release/broker/I386/EventBind.obj \
../Static_Release/broker/I386/EventBrokerLinkDown.obj \
../Static_Release/broker/I386/EventBrokerLinkUp.obj \
../Static_Release/broker/I386/EventClientConnect.obj \
../Static_Release/broker/I386/EventClientConnectFail.obj \
../Static_Release/broker/I386/EventClientDisconnect.obj \
../Static_Release/broker/I386/EventDeny.obj \
../Static_Release/broker/I386/EventExchangeDeclare.obj \
../Static_Release/broker/I386/EventExchangeDelete.obj \
../Static_Release/broker/I386/EventFileLoadFailed.obj \
../Static_Release/broker/I386/EventFileLoaded.obj \
../Static_Release/broker/I386/EventQueueDeclare.obj \
../Static_Release/broker/I386/EventQueueDelete.obj \
../Static_Release/broker/I386/EventSubscribe.obj \
../Static_Release/broker/I386/EventUnbind.obj \
../Static_Release/broker/I386/EventUnsubscribe.obj \
../Static_Release/broker/I386/Exchange.obj \
../Static_Release/broker/I386/Exchange2.obj \
../Static_Release/broker/I386/ExchangeRegistry.obj \
../Static_Release/broker/I386/FanOutExchange.obj \
../Static_Release/broker/I386/HeadersExchange.obj \
../Static_Release/broker/I386/IncompleteMessageList.obj \
../Static_Release/broker/I386/Link.obj \
../Static_Release/broker/I386/Link2.obj \
../Static_Release/broker/I386/LinkRegistry.obj \
../Static_Release/broker/I386/ManagementBroker.obj \
../Static_Release/broker/I386/ManagementExchange.obj \
../Static_Release/broker/I386/Message.obj \
../Static_Release/broker/I386/MessageAdapter.obj \
../Static_Release/broker/I386/MessageBuilder.obj \
../Static_Release/broker/I386/MessageStoreModule.obj \
../Static_Release/broker/I386/NameGenerator.obj \
../Static_Release/broker/I386/NullMessageStore.obj \
../Static_Release/broker/I386/Package.obj \
../Static_Release/broker/I386/Package2.obj \
../Static_Release/broker/I386/PersistableMessage.obj \
../Static_Release/broker/I386/QpiddBroker.obj \
../Static_Release/broker/I386/Queue.obj \
../Static_Release/broker/I386/Queue2.obj \
../Static_Release/broker/I386/QueueBindings.obj \
../Static_Release/broker/I386/QueueCleaner.obj \
../Static_Release/broker/I386/QueueListeners.obj \
../Static_Release/broker/I386/QueuePolicy.obj \
../Static_Release/broker/I386/QueueRegistry.obj \
../Static_Release/broker/I386/RateTracker.obj \
../Static_Release/broker/I386/RecoveredDequeue.obj \
../Static_Release/broker/I386/RecoveredEnqueue.obj \
../Static_Release/broker/I386/RecoveryManagerImpl.obj \
../Static_Release/broker/I386/SaslAuthenticator.obj \
../Static_Release/broker/I386/SemanticState.obj \
../Static_Release/broker/I386/Session.obj \
../Static_Release/broker/I386/SessionAdapter.obj \
../Static_Release/broker/I386/SessionHandler.obj \
../Static_Release/broker/I386/SessionManager.obj \
../Static_Release/broker/I386/SessionState.obj \
../Static_Release/broker/I386/System.obj \
../Static_Release/broker/I386/System2.obj \
../Static_Release/broker/I386/TCPIOPlugin.obj \

../Static_Release/broker/I386/Timer.obj \
../Static_Release/broker/I386/TopicExchange.obj \
../Static_Release/broker/I386/TxAccept.obj \
../Static_Release/broker/I386/TxBuffer.obj \
../Static_Release/broker/I386/TxPublish.obj \
../Static_Release/broker/I386/Vhost.obj \
../Static_Release/broker/I386/Vhost2.obj \
../Static_Release/broker/I386/qpidd.obj

qmfconsole_OBJECTS=../Static_Release/qmfconsole/I386/ClassKey.obj

lib:
lib.exe /OUT:"../qpidbrokers.lib" $(broker_OBJECTS)

5.4.2) tests.mk
C89_COMPILER=C:/Program Files (x86)/Microsoft Visual Studio 9.0/VC/bin/cl.exe
C89_LINKER=C:/Program Files (x86)/Microsoft Visual Studio 9.0/VC/bin/link.exe
LINK="link.exe"

OBJEXT=.obj
EXEEXT=.exe

INCLUDES=/I "C:/Program Files (x86)/boost/boost_1_36_0/include/103600" /I "C:/Program Files (x86)/boost/boost_1_36_0/." /I "." /I ".." /I "../gen"
FLAGS=/D "NDEBUG" /D "WIN32" /D "_CONSOLE" /D "_CRT_NONSTDC_NO_WARNINGS" /D "NOMINMAX" /D "WIN32_LEAN_AND_MEAN" /D
"_SCL_SECURE_NO_WARNINGS" /D "_CRT_SECURE_NO_WARNINGS" /FD /EHsc /MT /W3 /c /TP /wd4244 /wd4800 /wd4290 /wd4355

lib_client=..\qpidclients.lib
lib_common=..\qpidcommons.lib
lib_broker=..\qpidbrokers.lib
#lib_console=..\qmfconsoles.lib
LDLIBS=..\qpidclients.lib ..\qpidcommons.lib ..\qpidbrokers.lib
LIBS=ws2_32.lib secur32.lib rpcrt4.lib kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib
odbc32.lib odbccp32.lib
LDFLAGS=/INCREMENTAL:NO /LIBPATH:"." /LIBPATH:"C:\Program Files (x86)\boost\boost_1_36_0\lib" /DEBUG /SUBSYSTEM:CONSOLE /OPT:REF
/OPT:ICF /DYNAMICBASE /NXCOMPAT /MACHINE:X86 /nologo /errorReport:prompt

RELDIR=Static_Release
OBJDIR=$(RELDIR)/tests/I386
#UNIT_TEST_OBJDIR=$(RELDIR)/unit_test/I386
ROOTDIR=C:\Users\v-vinsea\Dev\AMQP\qpid-M4\cpp\src\tests

check_PROGRAMS=
check_LTLIBRARIES=
TESTS=
EXTRA_DIST=
CLEANFILES=

SOURCES=
UNIT_TEST_OBJECTS=

```
unit_test_SOURCES= \
ConsoleTest.cpp \
unit_test.cpp \
exception_test.cpp \
RefCounted.cpp \
SessionState.cpp Blob.cpp logging.cpp \
AsyncCompletion.cpp \
Url.cpp Uuid.cpp \
Shlib.cpp FieldValue.cpp FieldTable.cpp Array.cpp \
QueueOptionsTest.cpp \
InlineAllocator.cpp \
InlineVector.cpp \
ClientSessionTest.cpp \
SequenceSet.cpp \
StringUtils.cpp \
IncompleteMessageList.cpp \
RangeSet.cpp \
AtomicValue.cpp \
QueueTest.cpp \
AccumulatedAckTest.cpp \
DtxWorkRecordTest.cpp \
DeliveryRecordTest.cpp \
ExchangeTest.cpp \
HeadersExchangeTest.cpp \
MessageTest.cpp \
QueueRegistryTest.cpp \
QueuePolicyTest.cpp \
FramingTest.cpp \
HeaderTest.cpp \
SequenceNumberTest.cpp \
TimerTest.cpp \
TopicExchangeTest.cpp \
TxBufferTest.cpp \
TxPublishTest.cpp \
MessageBuilderTest.cpp \
ManagementTest.cpp \
MessageReplayTracker.cpp

perftest_SOURCES=perftest.cpp
txtest_SOURCES=txtest.cpp
latencytest_SOURCES=latencytest.cpp
echotest_SOURCES=echotest.cpp
client_test_SOURCES=client_test.cpp
topic_listener_SOURCES=topic_listener.cpp
topic_publisher_SOURCES=topic_publisher.cpp
publish_SOURCES=publish.cpp
consume_SOURCES=consume.cpp
header_test_SOURCES=header_test.cpp
failover_soak_SOURCES=failover_soak.cpp
declare_queues_SOURCES=declare_queues.cpp
replaying_sender_SOURCES=replaying_sender.cpp
resuming_receiver_SOURCES=resuming_receiver.cpp
txshift_SOURCES=txshift.cpp
txjob_SOURCES=txjob.cpp
receiver_SOURCES=receiver.cpp
sender_SOURCES=sender.cpp

SOURCES=$(SOURCES) $(unit_test_SOURCES)
SOURCES=$(SOURCES) $(perftest_SOURCES)
SOURCES=$(SOURCES) $(txtest_SOURCES)
SOURCES=$(SOURCES) $(latencytest_SOURCES)
SOURCES=$(SOURCES) $(echotest_SOURCES)
SOURCES=$(SOURCES) $(client_test_SOURCES)
SOURCES=$(SOURCES) $(topic_listener_SOURCES)
SOURCES=$(SOURCES) $(topic_publisher_SOURCES)
SOURCES=$(SOURCES) $(publish_SOURCES)
SOURCES=$(SOURCES) $(consume_SOURCES)
SOURCES=$(SOURCES) $(header_test_SOURCES)
#SOURCES=$(SOURCES) $(failover_soak_SOURCES)
SOURCES=$(SOURCES) $(declare_queues_SOURCES)
SOURCES=$(SOURCES) $(replaying_sender_SOURCES)
SOURCES=$(SOURCES) $(resuming_receiver_SOURCES)
SOURCES=$(SOURCES) $(txshift_SOURCES)
SOURCES=$(SOURCES) $(txjob_SOURCES)
SOURCES=$(SOURCES) $(receiver_SOURCES)
SOURCES=$(SOURCES) $(sender_SOURCES)
```

```
UNIT_TEST_OBJECTS= \
$(OBJDIR)/unit_test$(OBJEXT) \
$(OBJDIR)/exception_test$(OBJEXT) \
$(OBJDIR)/RefCounted$(OBJEXT) \
$(OBJDIR)/SessionState$(OBJEXT) \
$(OBJDIR)/Blob$(OBJEXT) \
$(OBJDIR)/logging$(OBJEXT) \
$(OBJDIR)/AsyncCompletion$(OBJEXT) \
$(OBJDIR)/Url$(OBJEXT) \
$(OBJDIR)/Uuid$(OBJEXT) \
$(OBJDIR)/Shlib$(OBJEXT) \
$(OBJDIR)/FieldValue$(OBJEXT) \
$(OBJDIR)/FieldTable$(OBJEXT) \
$(OBJDIR)/Array$(OBJEXT) \
$(OBJDIR)/QueueOptionsTest$(OBJEXT) \
$(OBJDIR)/InlineAllocator$(OBJEXT) \
$(OBJDIR)/InlineVector$(OBJEXT) \
$(OBJDIR)/ClientSessionTest$(OBJEXT) \
$(OBJDIR)/SequenceSet$(OBJEXT) \
$(OBJDIR)/StringUtils$(OBJEXT) \
$(OBJDIR)/IncompleteMessageList$(OBJEXT) \
$(OBJDIR)/RangeSet$(OBJEXT) \
$(OBJDIR)/AtomicValue$(OBJEXT) \
$(OBJDIR)/QueueTest$(OBJEXT) \
$(OBJDIR)/AccumulatedAckTest$(OBJEXT) \
$(OBJDIR)/DtxWorkRecordTest$(OBJEXT) \
$(OBJDIR)/DeliveryRecordTest$(OBJEXT) \
$(OBJDIR)/ExchangeTest$(OBJEXT) \
$(OBJDIR)/HeadersExchangeTest$(OBJEXT) \
$(OBJDIR)/MessageTest$(OBJEXT) \
$(OBJDIR)/QueueRegistryTest$(OBJEXT) \
$(OBJDIR)/QueuePolicyTest$(OBJEXT) \
$(OBJDIR)/FramingTest$(OBJEXT) \
$(OBJDIR)/HeaderTest$(OBJEXT) \
$(OBJDIR)/SequenceNumberTest$(OBJEXT) \
$(OBJDIR)/TimerTest$(OBJEXT) \
$(OBJDIR)/TopicExchangeTest$(OBJEXT) \
$(OBJDIR)/TxBufferTest$(OBJEXT) \
$(OBJDIR)/TxPublishTest$(OBJEXT) \
$(OBJDIR)/MessageBuilderTest$(OBJEXT) \
$(OBJDIR)/ManagementTest$(OBJEXT) \
$(OBJDIR)/MessageReplayTracker$(OBJEXT)
```

1. $(OBJDIR)/ConsoleTest$(OBJEXT) \

```
TESTS=$(TESTS) $(RELDIR)/unit_test$(EXEEXT)
TESTS=$(TESTS) $(RELDIR)/perftest$(EXEEXT)
TESTS=$(TESTS) $(RELDIR)/txtest$(EXEEXT)
TESTS=$(TESTS) $(RELDIR)/latencytest$(EXEEXT)
TESTS=$(TESTS) $(RELDIR)/echotest$(EXEEXT)
TESTS=$(TESTS) $(RELDIR)/client_test$(EXEEXT)
TESTS=$(TESTS) $(RELDIR)/topic_listener$(EXEEXT)
TESTS=$(TESTS) $(RELDIR)/topic_publisher$(EXEEXT)
TESTS=$(TESTS) $(RELDIR)/publish$(EXEEXT)
TESTS=$(TESTS) $(RELDIR)/consume$(EXEEXT)
TESTS=$(TESTS) $(RELDIR)/header_test$(EXEEXT)
#TESTS=$(TESTS) $(RELDIR)/failover_soak$(EXEEXT)
TESTS=$(TESTS) $(RELDIR)/declare_queues$(EXEEXT)
TESTS=$(TESTS) $(RELDIR)/replaying_sender$(EXEEXT)
TESTS=$(TESTS) $(RELDIR)/resuming_receiver$(EXEEXT)
TESTS=$(TESTS) $(RELDIR)/txshift$(EXEEXT)
TESTS=$(TESTS) $(RELDIR)/txjob$(EXEEXT)
TESTS=$(TESTS) $(RELDIR)/receiver$(EXEEXT)
TESTS=$(TESTS) $(RELDIR)/sender$(EXEEXT)

.cpp.obj:
@mkdir -p $(RELDIR)
@mkdir -p $(OBJDIR)
$(CC) /O2 $(INCLUDES) $(FLAGS) /Fo"$(OBJDIR)//" /Fd"$(OBJDIR)/vc90.pdb" $**

all: $(SOURCES:.cpp=.obj) $(TESTS)

$(RELDIR)/perftest$(EXEEXT): {$(OBJDIR)}$(perftest_SOURCES:.cpp=.obj)
$(LINK) /OUT:$(RELDIR)/%|fF$(EXEEXT) /PDB:$(ROOTDIR)/$(RELDIR)/%|fF.pdb /MANIFEST /MANIFESTFILE:"$(OBJDIR)%|fF$(EXEEXT).
intermediate.manifest" /MANIFESTUAC:"level='asInvoker' uiAccess='false'" $(LDFLAGS) $(LDLIBS) $(LIBS) %|pfF$(OBJEXT)

$(RELDIR)/txtest$(EXEEXT): {$(OBJDIR)}$(txtest_SOURCES:.cpp=.obj)
$(LINK) /OUT:$(RELDIR)/%|fF$(EXEEXT) /PDB:$(ROOTDIR)/$(RELDIR)/%|fF.pdb /MANIFEST /MANIFESTFILE:"$(OBJDIR)%|fF$(EXEEXT).
intermediate.manifest" /MANIFESTUAC:"level='asInvoker' uiAccess='false'" $(LDFLAGS) $(LDLIBS) $(LIBS) %|pfF$(OBJEXT)
```

```
$(RELDIR)/latencytest$(EXEEXT): {$(OBJDIR)}$(latencytest_SOURCES:.cpp=.obj)
$(LINK) /OUT:$(RELDIR)/%|fF$(EXEEXT) /PDB:$(ROOTDIR)/$(RELDIR)/%|fF.pdb /MANIFEST /MANIFESTFILE:"$(OBJDIR)%|fF$(EXEEXT).
intermediate.manifest" /MANIFESTUAC:"level='asInvoker' uiAccess='false'" $(LDFLAGS) $(LDLIBS) $(LIBS) %|pfF$(OBJEXT)

$(RELDIR)/echotest$(EXEEXT): {$(OBJDIR)}$(echotest_SOURCES:.cpp=.obj)
$(LINK) /OUT:$(RELDIR)/%|fF$(EXEEXT) /PDB:$(ROOTDIR)/$(RELDIR)/%|fF.pdb /MANIFEST /MANIFESTFILE:"$(OBJDIR)%|fF$(EXEEXT).
intermediate.manifest" /MANIFESTUAC:"level='asInvoker' uiAccess='false'" $(LDFLAGS) $(LDLIBS) $(LIBS) %|pfF$(OBJEXT)

$(RELDIR)/client_test$(EXEEXT): {$(OBJDIR)}$(client_test_SOURCES:.cpp=.obj)
$(LINK) /OUT:$(RELDIR)/%|fF$(EXEEXT) /PDB:$(ROOTDIR)/$(RELDIR)/%|fF.pdb /MANIFEST /MANIFESTFILE:"$(OBJDIR)%|fF$(EXEEXT).
intermediate.manifest" /MANIFESTUAC:"level='asInvoker' uiAccess='false'" $(LDFLAGS) $(LDLIBS) $(LIBS) %|pfF$(OBJEXT)

$(RELDIR)/topic_listener$(EXEEXT): {$(OBJDIR)}$(topic_listener_SOURCES:.cpp=.obj)
$(LINK) /OUT:$(RELDIR)/%|fF$(EXEEXT) /PDB:$(ROOTDIR)/$(RELDIR)/%|fF.pdb /MANIFEST /MANIFESTFILE:"$(OBJDIR)%|fF$(EXEEXT).
intermediate.manifest" /MANIFESTUAC:"level='asInvoker' uiAccess='false'" $(LDFLAGS) $(LDLIBS) $(LIBS) %|pfF$(OBJEXT)

$(RELDIR)/topic_publisher$(EXEEXT): {$(OBJDIR)}$(topic_publisher_SOURCES:.cpp=.obj)
$(LINK) /OUT:$(RELDIR)/%|fF$(EXEEXT) /PDB:$(ROOTDIR)/$(RELDIR)/%|fF.pdb /MANIFEST /MANIFESTFILE:"$(OBJDIR)%|fF$(EXEEXT).
intermediate.manifest" /MANIFESTUAC:"level='asInvoker' uiAccess='false'" $(LDFLAGS) $(LDLIBS) $(LIBS) %|pfF$(OBJEXT)

$(RELDIR)/publish$(EXEEXT): {$(OBJDIR)}$(publish_SOURCES:.cpp=.obj)
$(LINK) /OUT:$(RELDIR)/%|fF$(EXEEXT) /PDB:$(ROOTDIR)/$(RELDIR)/%|fF.pdb /MANIFEST /MANIFESTFILE:"$(OBJDIR)%|fF$(EXEEXT).
intermediate.manifest" /MANIFESTUAC:"level='asInvoker' uiAccess='false'" $(LDFLAGS) $(LDLIBS) $(LIBS) %|pfF$(OBJEXT)

$(RELDIR)/consume$(EXEEXT): {$(OBJDIR)}$(consume_SOURCES:.cpp=.obj)
$(LINK) /OUT:$(RELDIR)/%|fF$(EXEEXT) /PDB:$(ROOTDIR)/$(RELDIR)/%|fF.pdb /MANIFEST /MANIFESTFILE:"$(OBJDIR)%|fF$(EXEEXT).
intermediate.manifest" /MANIFESTUAC:"level='asInvoker' uiAccess='false'" $(LDFLAGS) $(LDLIBS) $(LIBS) %|pfF$(OBJEXT)

$(RELDIR)/header_test$(EXEEXT): {$(OBJDIR)}$(header_test_SOURCES:.cpp=.obj)
$(LINK) /OUT:$(RELDIR)/%|fF$(EXEEXT) /PDB:$(ROOTDIR)/$(RELDIR)/%|fF.pdb /MANIFEST /MANIFESTFILE:"$(OBJDIR)%|fF$(EXEEXT).
intermediate.manifest" /MANIFESTUAC:"level='asInvoker' uiAccess='false'" $(LDFLAGS) $(LDLIBS) $(LIBS) %|pfF$(OBJEXT)

$(RELDIR)/failover_soak$(EXEEXT): {$(OBJDIR)}$(failover_soak_SOURCES:.cpp=.obj)
$(LINK) /OUT:$(RELDIR)/%|fF$(EXEEXT) /PDB:$(ROOTDIR)/$(RELDIR)/%|fF.pdb /MANIFEST /MANIFESTFILE:"$(OBJDIR)%|fF$(EXEEXT).
intermediate.manifest" /MANIFESTUAC:"level='asInvoker' uiAccess='false'" $(LDFLAGS) $(LDLIBS) $(LIBS) %|pfF$(OBJEXT)

$(RELDIR)/declare_queues$(EXEEXT): {$(OBJDIR)}$(declare_queues_SOURCES:.cpp=.obj)
$(LINK) /OUT:$(RELDIR)/%|fF$(EXEEXT) /PDB:$(ROOTDIR)/$(RELDIR)/%|fF.pdb /MANIFEST /MANIFESTFILE:"$(OBJDIR)%|fF$(EXEEXT).
intermediate.manifest" /MANIFESTUAC:"level='asInvoker' uiAccess='false'" $(LDFLAGS) $(LDLIBS) $(LIBS) %|pfF$(OBJEXT)

$(RELDIR)/replaying_sender$(EXEEXT): {$(OBJDIR)}$(replaying_sender_SOURCES:.cpp=.obj)
$(LINK) /OUT:$(RELDIR)/%|fF$(EXEEXT) /PDB:$(ROOTDIR)/$(RELDIR)/%|fF.pdb /MANIFEST /MANIFESTFILE:"$(OBJDIR)%|fF$(EXEEXT).
intermediate.manifest" /MANIFESTUAC:"level='asInvoker' uiAccess='false'" $(LDFLAGS) $(LDLIBS) $(LIBS) %|pfF$(OBJEXT)

$(RELDIR)/resuming_receiver$(EXEEXT): {$(OBJDIR)}$(resuming_receiver_SOURCES:.cpp=.obj)
$(LINK) /OUT:$(RELDIR)/%|fF$(EXEEXT) /PDB:$(ROOTDIR)/$(RELDIR)/%|fF.pdb /MANIFEST /MANIFESTFILE:"$(OBJDIR)%|fF$(EXEEXT).
intermediate.manifest" /MANIFESTUAC:"level='asInvoker' uiAccess='false'" $(LDFLAGS) $(LDLIBS) $(LIBS) %|pfF$(OBJEXT)

$(RELDIR)/txshift$(EXEEXT): {$(OBJDIR)}$(txshift_SOURCES:.cpp=.obj)
$(LINK) /OUT:$(RELDIR)/%|fF$(EXEEXT) /PDB:$(ROOTDIR)/$(RELDIR)/%|fF.pdb /MANIFEST /MANIFESTFILE:"$(OBJDIR)%|fF$(EXEEXT).
intermediate.manifest" /MANIFESTUAC:"level='asInvoker' uiAccess='false'" $(LDFLAGS) $(LDLIBS) $(LIBS) %|pfF$(OBJEXT)

$(RELDIR)/txjob$(EXEEXT): {$(OBJDIR)}$(txjob_SOURCES:.cpp=.obj)
$(LINK) /OUT:$(RELDIR)/%|fF$(EXEEXT) /PDB:$(ROOTDIR)/$(RELDIR)/%|fF.pdb /MANIFEST /MANIFESTFILE:"$(OBJDIR)%|fF$(EXEEXT).
intermediate.manifest" /MANIFESTUAC:"level='asInvoker' uiAccess='false'" $(LDFLAGS) $(LDLIBS) $(LIBS) %|pfF$(OBJEXT)

$(RELDIR)/receiver$(EXEEXT): {$(OBJDIR)}$(receiver_SOURCES:.cpp=.obj)
$(LINK) /OUT:$(RELDIR)/%|fF$(EXEEXT) /PDB:$(ROOTDIR)/$(RELDIR)/%|fF.pdb /MANIFEST /MANIFESTFILE:"$(OBJDIR)%|fF$(EXEEXT).
intermediate.manifest" /MANIFESTUAC:"level='asInvoker' uiAccess='false'" $(LDFLAGS) $(LDLIBS) $(LIBS) %|pfF$(OBJEXT)

$(RELDIR)/sender$(EXEEXT): {$(OBJDIR)}$(sender_SOURCES:.cpp=.obj)
$(LINK) /OUT:$(RELDIR)/%|fF$(EXEEXT) /PDB:$(ROOTDIR)/$(RELDIR)/%|fF.pdb /MANIFEST /MANIFESTFILE:"$(OBJDIR)%|fF$(EXEEXT).
intermediate.manifest" /MANIFESTUAC:"level='asInvoker' uiAccess='false'" $(LDFLAGS) $(LDLIBS) $(LIBS) %|pfF$(OBJEXT)

$(RELDIR)/unit_test$(EXEEXT): {$(OBJDIR)}$(unit_test_SOURCES:.cpp=.obj)
$(LINK) /OUT:$(RELDIR)/%|fF$(EXEEXT) /PDB:$(ROOTDIR)/$(RELDIR)/%|fF.pdb /MANIFEST /MANIFESTFILE:"$(OBJDIR)%|fF$(EXEEXT).
intermediate.manifest" /MANIFESTUAC:"level='asInvoker' uiAccess='false'" $(LDFLAGS) $(LDLIBS) $(LIBS) $(UNIT_TEST_OBJECTS)
```