How to build your application with FIXAntenna C++ on Linux

C++ ABI references

- https://gcc.gnu.org/onlinedocs/libstdc++/manual/abi.html
- https://gcc.gnu.org/onlinedocs/libstdc++/manual/api.html
- https://gcc.gnu.org/onlinedocs/libstdc++/manual/using_dual_abi.html
- https://www.gnu.org/software/gcc/projects/cxx-status.html

Key notes

- GCC C++ ABI is forward compatible from version 3.4.
- GCC 5.1 introduced Dual ABI to support C++11 requirements for `std::string` and `std::list` (macro `_GLIBCXX_USE_CXX11_ABI` specifies which one to use).
- Since GCC 5.1 C++11 ABI is compatible with GCC 5.x and above where C++ standard is 2011 and newer.
- GCC 6.1 has full support for the C++14 standard, which was published in 2014. This mode is the default in GCC 6.1 and above; it can be explicitly selected with the `-std=c++14.`

The previous naming schema was ambiguous because gcc 5 and later can be used to build binaries with C++98 ABI.
Since the release 2.26.0 FIX Antenna binaries have following naming convention for Linux:

Binary	Old binary naming convention	C++ ABI	Target OS	Target Compiler
`libV12_rh6.so`	libV12-centos-6.5-gcc44-MD-x64.so	C++ 98	RHEL6 /CentOS6	1. GCC 4.4 - 4.9 2. GCC 5.2 and higher with `-D_GLIBCXX_USE_CXX11_ABI=0`, any C++ standard supported by the compiler can be used (C++ 98, 11, 14, 17). **Note:** GCC should be installed from RHEL/CentOS repository or developer toolset.
`libV12_rh6_cxx11.so`	libV12-centos-6.7-gcc52-MD-x64.so libV12-centos-6.10-gcc52-MD-x64.so libV12-centos-6.9-gcc63-MD-x64.so libV12-centos-6.10-gcc63-MD-x64.so	C++ 11	RHEL6 /CentOS6	GCC 5.2 and higher with `-D_GLIBCXX_USE_CXX11_ABI=1` (the default), any C++ standard supported by the compiler can be used (C++ 98, 11, 14, 17). **Note:** GCC 5.2 and higher should be build and installed from sources, because GCC from devtoolset uses only C++98 ABI. Use LD version 2.27 and higher with C++11 ABI to avoid linking problems.
`libV12_rh7.so`	libV12-centos-7.5.1804-gcc48-MD-x64.so libV12-centos-7.5.1804-gcc49-MD-x64.so	C++ 98	RHEL7 /CentOS7	1. GCC 4.8 - 4.9 2. GCC 5.2 and higher with `-D_GLIBCXX_USE_CXX11_ABI=0`, any C++ standard supported by the compiler can be used (C++ 98, 11, 14, 17). **Note:** GCC should be installed from RHEL/CentOS repository or developer toolset. Do not use GCC from devtoolset with new ABI, i.e. with parameter `-D_GLIBCXX_USE_CXX11_ABI=1. This compiler build uses C++98 ABI.`
`libV12_rh7_cxx11.so`	libV12-centos-7.5.1804-gcc52-MD-x64.so	C++ 11	RHEL7 /CentOS7	GCC 5.2 and higher with `-D_GLIBCXX_USE_CXX11_ABI=1` (the default), any C++ standard supported by the compiler can be used (C++ 98, 11, 14, 17). **Note:** GCC 5.2 and higher should be build and installed from sources, because GCC from devtoolset uses only C++98 ABI. Use LD version 2.27 and higher with C++11 ABI to avoid linking problems.
`libV12_ubuntu1604_cxx11.so`	libV12-ubuntu-16.04.5-lts-gcc61-MD-x64.so	C++ 11	Ubuntu 16.04	GCC 5.2 and higher with `-D_GLIBCXX_USE_CXX11_ABI=1` (the default), any C++ standard supported by the compiler can be used (C++ 98, 11, 14, 17). **Note:** GCC 5.2 and higher should be installed from Ubuntu repository. Use LD version 2.27 and higher with C++11 ABI to avoid linking problems.

How to use FIX Antenna with GCC

Preparation step: set symlink to suitable binary for your platform. Use *${FIX_ANTENNA_LOCATION}/samples/v12-linker.sh* helper for that purpose or use *lr* from binutils.

How to set symlink to libV12.so

```
# FOR RHEL7/CentOS7
# C++98 ABI gcc 4.4-4.9 and gcc 5.2 and higher with -D_GLIBCXX_USE_CXX11_ABI=0
$ ./v12-linker.sh --use-cxx11-abi=0 --searchpath=../lib
[INFO] libV12_rh7.so is used as libV12.so


# is equal to
ln -fs libV12_rh7.so libV12.so


# FOR RHEL7/CentOS7
# C++11 ABI, custom build gcc 5.2 and higher with -D_GLIBCXX_USE_CXX11_ABI=1(by default)
$ ./v12-linker.sh --use-cxx11-abi=1 --searchpath=../lib
[INFO] libV12_rh7_cxx11.so is used as libV12.so

# is equal to
ln -fs libV12_rh7_cxx11.so libV12.so
```

Take *${FIX_ANTENNA_LOCATION}/samples/FIX_Quickstart* example for this tutorial.

First way: build the project using the parameters of the compiler and linker from scratch.

Second way: build with make - a build automation tool for compiling and linking applications.
The project has following structure:

```
[user@fa-devenv FIX_QuickStart]$ tree -p Listener/ Sender/
Listener/
├── [-rw-rw-r--]  cert.pem
├── [-rwxr-xr-x]  clean.sh
├── [drwxrwxr-x]  debug_obj
├── [-rw-rw-r--]  FIXApp.cpp
├── [-rw-rw-r--]  FIXApp.h
├── [-rw-rw-r--]  GNUmakefile.debug
├── [-rw-rw-r--]  GNUmakefile.release
├── [-rw-rw-r--]  Listener.cpp
├── [-rw-rw-r--]  listener.engine.properties
├── [drwxrwxr-x]  logs
│   └── [drwxrwxr-x]  backup
├── [drwxrwxr-x]  release_obj
├── [-rwxr-xr-x]  runD.sh
├── [-rwxr-xr-x]  run.sh
├── [-rw-rw-r--]  SessionConnector.cpp
└── [-rw-rw-r--]  SessionConnector.h
Sender/
├── [-rwxr-xr-x]  clean.sh
├── [drwxrwxr-x]  debug_obj
├── [-rw-rw-r--]  FIXApp.cpp
├── [-rw-rw-r--]  FIXApp.h
├── [-rw-rw-r--]  GNUmakefile.debug
├── [-rw-rw-r--]  GNUmakefile.release
├── [drwxrwxr-x]  logs
│   └── [drwxrwxr-x]  backup
├── [drwxrwxr-x]  release_obj
├── [-rwxr-xr-x]  runD.sh
├── [-rwxr-xr-x]  run.sh
├── [-rw-rw-r--]  Sender.cpp
└── [-rw-rw-r--]  sender.engine.properties

8 directories, 21 files
[user@fa-devenv FIX_QuickStart]$
```

Go to the FIX Message Listener directory and compile it with following parameters:

Compile Listener

```
cd Listener

g++ -D_LINUX -fPIC -MMD -MF ./debug_obj/FIXApp.d -I../../../headers -ggdb -D_DEBUG -c FIXApp.cpp -o debug_obj
/FIXApp.o
g++ -D_LINUX -fPIC -MMD -MF ./debug_obj/Listener.d -I../../../headers -ggdb -D_DEBUG -c Listener.cpp -o
debug_obj/Listener.o
g++ -D_LINUX -fPIC -MMD -MF ./debug_obj/SessionConnector.d -I../../../headers -ggdb -D_DEBUG -c
SessionConnector.cpp -o debug_obj/SessionConnector.o
```

Compile with optional flags

```
# or with optional warnings

g++ -D_LINUX -Wall -Wextra -Winit-self -Wmissing-include-dirs -Wno-parentheses -Wpedantic \
-Wno-long-long -Wno-unused-parameter -Wredundant-decls -Wnon-virtual-dtor -Wshadow -Woverloaded-virtual \
-fPIC -MMD -MF ./debug_obj/FIXApp.d -I../../../headers -ggdb -D_DEBUG -c FIXApp.cpp -o debug_obj/FIXApp.o

g++ -D_LINUX -Wall -Wextra -Winit-self -Wmissing-include-dirs -Wno-parentheses -Wpedantic \
 -Wno-long-long -Wno-unused-parameter -Wredundant-decls -Wnon-virtual-dtor -Wshadow -Woverloaded-virtual \
 -fPIC -MMD -MF ./debug_obj/Listener.d -I../../../headers -ggdb -D_DEBUG -c Listener.cpp -o debug_obj/Listener.o

g++ -D_LINUX -Wall -Wextra -Winit-self -Wmissing-include-dirs -Wno-parentheses -Wpedantic \
 -Wno-long-long -Wno-unused-parameter -Wredundant-decls -Wnon-virtual-dtor -Wshadow -Woverloaded-virtual \
 -fPIC -MMD -MF ./debug_obj/SessionConnector.d -I../../../headers -ggdb -D_DEBUG -c SessionConnector.cpp -o
debug_obj/SessionConnector.o
```

Do the same things for Sender:

Compile Sender

```
cd ../Sender

g++ -D_LINUX -fPIC -MMD -MF ./debug_obj/FIXApp.d -I../../../headers -ggdb -D_DEBUG -c FIXApp.cpp -o debug_obj
/FIXApp.o
g++ -D_LINUX -fPIC -MMD -MF ./debug_obj/Sender.d -I../../../headers -ggdb -D_DEBUG -c Sender.cpp -o debug_obj
/Sender.o
```

Compile with optional flags

```
# or with optional warnings

g++ -D_LINUX -Wall -Wextra -Winit-self -Wmissing-include-dirs -Wno-parentheses -Wpedantic \
 -Wno-long-long -Wno-unused-parameter -Wredundant-decls -Wnon-virtual-dtor -Wshadow -Woverloaded-virtual \
 -fPIC -MMD -MF ./debug_obj/FIXApp.d -I../../../headers -ggdb -D_DEBUG -c FIXApp.cpp -o debug_obj/FIXApp.o

g++ -D_LINUX -Wall -Wextra -Winit-self -Wmissing-include-dirs -Wno-parentheses -Wpedantic \
 -Wno-long-long -Wno-unused-parameter -Wredundant-decls -Wnon-virtual-dtor -Wshadow -Woverloaded-virtual \
 -fPIC -MMD -MF ./debug_obj/Sender.d -I../../../headers -ggdb -D_DEBUG -c Sender.cpp -o debug_obj/Sender.o
```

Sender and Listener are compiled for debug mode. If you want to do it for release, use ***-DNDEBUG*** instead of ***-D_DEBUG***, remove ***-ggdb*** and add ***-O3*** flag to optimize the code.

*Please, pay attention at using **-I** flag to set location of FIX Antenna include directory (-I../../../headers).*
*Don't forget to set libV12.so location with **-L** flag (-L../../../lib) and to add FIX Antenna library as additional dependency with flag **-l** (-lV12) when you link executable:*

Run linker for debug mode

```
g++ -L../../../lib debug_obj/FIXApp.o debug_obj/Listener.o debug_obj/SessionConnector.o \
    -o ListenerD \
    -lrt -lV12
```

Run linker for release mode

```
g++ -L../../../lib release_obj/FIXApp.o release_obj/Listener.o release_obj/SessionConnector.o \
         -o Listener \
        -lrt -lV12
```

How to build project with FIX Antenna using Make utility

First, create Makefile with following content:

samples/FIX_Quickstart/Makefile

```
#
# Makefile
#
# $Id: Makefile,v 1.3 2007/12/06 15:51:27 sam Exp $


###############################################
# global and compiler related settings

ROOTDIR:=$(shell pwd)

CFLAGS+=-ggdb
CXXFLAGS+=-D_LINUX \
        -Wall \
        -Wextra \
        -Winit-self \
        -Wmissing-include-dirs \
        -Wno-parentheses \
        -pedantic \
        -Wno-long-long \
        -Wno-unused-parameter \
        -Wredundant-decls \
        -Wnon-virtual-dtor \
        -Wshadow \
        -Woverloaded-virtual
LIBS+=-lV12

DFLAGS=-MMD -MF $$(addprefix $$(OBJDIR)/,$$(notdir $$(<:%.cpp=%.d)))
SHROFLAGS=-fPIC
SOFLAGS=-shared
DEBUG_FLAGS=-ggdb -D_DEBUG
OPTIM_FLAGS=-O3 -DNDEBUG

###############################################
#try to guess platform

###############################################
# more beautiful output
#MAKEFLAGS+=--no-print-directory
Q=@

QCC:=@echo "(C) $$(notdir $$<)" ; $(CC)
QCXX:=@echo "(C++) $$(notdir $$<)" ; $(CXX)
QLNK:=@echo "(LINK) $$(notdir $$<)" ; $(CXX)
QAR:=@echo "(AR) $$(notdir $$<)" ; $(AR)
QMAKE:=@echo "(MAKE) $@" ; $(MAKE)
###############################################

export

SUBDIRS = \
        Listener\
        Sender

all debug clean::
    @if [ -x ../v12-linker.sh ]; then ../v12-linker.sh --use-cxx11-abi=`../check_abi.sh` --searchpath=../..
/lib; fi; \
    for i in $(SUBDIRS); do \
        ( cd $$i && make -f GNUmakefile.debug $@) || exit 1; \
    done

release clean::
    @if [ -x ../v12-linker.sh ]; then ../v12-linker.sh --use-cxx11-abi=`../check_abi.sh` --searchpath=../..
/lib; fi; \
    for i in $(SUBDIRS); do \
        ( cd $$i && make -f GNUmakefile.release $@ ) || exit 1; \
    done
```

GNUmakefile.debug and GNUmakefile.release files are created for debug and release configuration:

samples/FIX_Quickstart/Listener/GNUmakefile.debug

```
SRCDIR=.
OBJDIR=./debug_obj
OUTBINDIR=./
LDFLAGS+=-L../../../lib

SRCS=$(wildcard $(SRCDIR)/*.cpp)
OBJS=$(SRCS:$(SRCDIR)/%.cpp=$(OBJDIR)/%.o)
DEPS=$(SRCS:$(SRCDIR)/%.cpp=$(OBJDIR)/%.d)

INCLUDE_PATH=-I../../../headers
CXXFLAGS+=$(SHROFLAGS) $(DFLAGS) $(INCLUDE_PATH) $(DEBUG_FLAGS)
LIBS:= -lrt $(LIBS)

#debug
EXECUTABLE=$(OUTBINDIR)/ListenerD

-include $(ROOTDIR)/Makefile.incl

.PHONY: clean

all debug: $(OUTBINDIR) $(OBJDIR) $(EXECUTABLE)

$(EXECUTABLE): $(OBJS)
	$(CXX) $(LDFLAGS) $^ -o $@ $(LIBS)

-include $(DEPS)

$(OBJS): $(OBJDIR)/%.o: $(SRCDIR)/%.cpp
	$(CXX) $(CXXFLAGS) -c $< -o $@

$(OBJDIR):
	mkdir $@

$(OUTBINDIR):
	mkdir $@

clean:
	$(RM) $(OBJS)
	$(RM) $(DEPS)
```

samples/FIX_Quickstart/Listener/GNUmakefile.release

```
SRCDIR=.
OBJDIR=./release_obj
OUTBINDIR=./
LDFLAGS+=-L../../../lib

SRCS=$(wildcard $(SRCDIR)/*.cpp)
OBJS=$(SRCS:$(SRCDIR)/%.cpp=$(OBJDIR)/%.o)
DEPS=$(SRCS:$(SRCDIR)/%.cpp=$(OBJDIR)/%.d)

INCLUDE_PATH=-I../../../headers
CXXFLAGS+=$(SHROFLAGS) $(DFLAGS) $(INCLUDE_PATH) $(OPTIM_FLAGS)
LIBS:= -lrt $(LIBS)

EXECUTABLE=$(OUTBINDIR)/Listener

-include $(ROOTDIR)/Makefile.incl

.PHONY: clean

release: $(OUTBINDIR) $(OBJDIR) $(EXECUTABLE)

$(EXECUTABLE): $(OBJS)
	$(CXX) $(LDFLAGS) $^ -o $@ $(LIBS)

-include $(DEPS)

$(OBJS): $(OBJDIR)/%.o: $(SRCDIR)/%.cpp
	$(CXX) $(CXXFLAGS) -c $< -o $@

$(OBJDIR):
	mkdir $@

$(OUTBINDIR):
	mkdir $@

clean:
	$(RM) $(OBJS)
	$(RM) $(DEPS)
```

Go to parent directory FIX_Quickstart and run *make -j$(nproc)*. As result, Listener and Sender directories have the following structure:

```
[user@fa-devenv FIX_QuickStart]$ tree
.
├── Listener
│   ├── cert.pem
│   ├── clean.sh
│   ├── debug_obj
│   ├── FIXApp.cpp
│   ├── FIXApp.h
│   ├── GNUmakefile.debug
│   ├── GNUmakefile.release
│   ├── Listener
│   ├── Listener.cpp
│   ├── ListenerD
│   ├── listener.engine.properties
│   ├── logs
│   │   └── backup
│   ├── runD.sh
│   ├── run.sh
│   ├── SessionConnector.cpp
│   └── SessionConnector.h
├── Makefile
├── Makefile.incl
└── Sender
    ├── clean.sh
    ├── debug_obj
    ├── FIXApp.cpp
    ├── FIXApp.h
    ├── GNUmakefile.debug
    ├── GNUmakefile.release
    ├── logs
    │   └── backup
    ├── runD.sh
    ├── run.sh
    ├── Sender
    ├── Sender.cpp
    ├── SenderD
    └── sender.engine.properties
```

Sender output

```
[user@fa-devenv Sender]$ ./run.sh
starting FIX Sender...
starting FIX Sender session...
sending the message: 8=FIX.4.4|9=81|35=D|11=20190117-00:39:17|55=MSFT|54=1|60=20190117-00:39:
17|38=400|40=2|44=11.32|10=105|

Logon received for the session TargetCompIDSenderCompID:8=FIX.4.4
|9=80|35=A|49=SenderCompID|56=TargetCompID|34=11|52=20190117-00:39:17.868|98=0|108=30|10=062|

Message in session TargetCompIDSenderCompID
8=FIX.4.4|9=226|35=8|49=SenderCompID|56=TargetCompID|34=10|43=Y|52=20190117-00:39:17.872|122=20190117-00:36:
32.854|37=1|11=20190117-00:36:32|17=2|150=F|39=2|55=MSFT|54=1|38=400|32=400|31=11.32|151=0|14=400|6=11.
32|60=20190117-00:36:32|1056=1|10=164|
```

Listener output

```
[user@fa-devenv Listener]$ ./run.sh
starting FIX Listener...
starting FIX Listener session...
Session has been created. Listening for incoming connection.
Type 'exit' to exit > Logon received for the session SenderCompIDTargetCompID:8=FIX.4.4
|9=80|35=A|49=TargetCompID|56=SenderCompID|34=10|52=20190117-00:39:17.866|98=0|108=30|10=059|

Message in session SenderCompIDTargetCompID
8=FIX.4.4|9=175|35=D|49=TargetCompID|56=SenderCompID|34=11|43=Y|52=20190117-00:39:17.871|122=20190117-00:39:
17.866|11=20190117-00:39:17|55=MSFT|54=1|60=20190117-00:39:17|38=400|40=2|44=11.32|10=226|
```