Softwareschneiderei

Conan packaging for cppTango

Marius Elvert

Softwareschneiderei GmbH

My initial problem

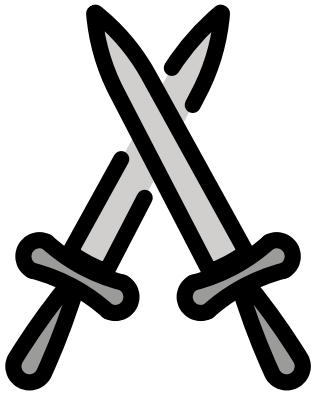
- Customer needed a new feature in a windows tango device server
- Hardware drivers only for windows
- Last built with an ancient version of Visual Studio (2005?)
- People were trading working binaries on USB sticks in the hallway

Dependency management in C++

- Extra hard in C++
  - OS, compiler, flags produce an, for the most part, unportable binary
- Use what your system provides
  - Need snow-flake build machines/docker/VMs
- Bundle dependencies with your project
  - Build with your project?
  - Bundle binaries?
  - Header only?

Why Conan?

- Gave me a „Conan the Frogarian“ T-Shirt
- Seems to be the most popular choice along with vcpkg
- Active community, lots of dependencies already available
- Open-source
- Decentralized
- Python
- Solves all my problems!



How does it work?

- Remotes always host „recipes“ to build from source.
  Adaptor around cmake/make/msbuild/scripts
- Optionally pre-build binaries.
- Binaries are stored locally in a cache.
  Cache acts as a „local server“, no network required.
- Select dependencies via `name/version[@user/channel]`
  e.g. `cpptango/9.3.3@softwareschneiderei/stable`
- Can consume from any build environment, though CMake is the most common

Configuration

- Binaries are matched to your project depending on settings and options, which can be bundled in a profile
- settings: project-global ABI-compatibility preferences like compiler and its version, OS, runtime, standard library, architecture
- options: library specific settings like `zlib:shared=True` to use an .so/.dll version of zlib

A device-server on Windows

- Sample server project, build with CMake.
  Assume you already have Visual Studio.
- Generated DS files with pogo.
- Install python and conan (via pip)
  https://docs.conan.io/en/latest/installation.html
- Packaged cppTango 9.3.3, so we're going to use that.
- One time: Add our conan remote:

```
conan remote add schneide https://api.bintray.com/conan/softwareschneiderei/conan
```

Basic CMakeLists.txt

```
1  project(TangoConanSample)
2
3  add_executable(ConanSample
4    ConanSample.cpp
5    ConanSample.h
6    ConanSampleClass.cpp
7    ConanSampleClass.h
8    ConanSampleStateMachine.cpp
9    main.cpp)
10
11 target_include_directories(ConanSample
12   PRIVATE .)
```

conanfile.txt

```
1  [requires]
2  cpptango/9.3.3@softwareschneiderei/stable
3
4  [generators]
5  cmake
```

- `requires` section:
  Your dependencies
- `generators` section:
  How to consume what conan gives you, e.g. your build system(s)

Install dependencies

- Go to your build folder and point conan at the source folder, e.g.:
  `conan install C:/Code/TangoConanSample`
- You can specify the settings with additional params:
  `-s build_type=Debug -s arch=x86`
- This will download your dependencies and generate a file with all the paths to it: `conanbuildinfo.cmake`

```
D:\ConanTangoSample\build\x64-Release>conan install C:\Code\TangoConanSample
Configuration:
[settings]
arch=x86_64
arch_build=x86_64
build_type=Release
compiler=Visual Studio
compiler.runtime=MD
compiler.version=16
os=Windows
os_build=Windows
```

Integrate into your CMakeLists.txt

```
1   cmake_minimum_required(VERSION 3.11)
2   project(TangoConanSample)
3
4   if (NOT EXISTS ${CMAKE_BINARY_DIR}/conanbuildinfo.cmake)
5     message(FATAL_ERROR "Please run 'conan install' first")
6   endif()
7   include(${CMAKE_BINARY_DIR}/conanbuildinfo.cmake)
8   conan_basic_setup(TARGETS)
9
10  add_executable(ConanSample
11    ConanSample.cpp
12    ConanSample.h
13    ConanSampleClass.cpp
14    ConanSampleClass.h
15    ConanSampleStateMachine.cpp
16    main.cpp)
17
18  target_include_directories(ConanSample
19    PRIVATE .)
20
21  target_link_libraries(ConanSample
22    PUBLIC CONAN_PKG::cpptango)
23
```

And it compiles!

```
>------ Build All started: Project: TangoConanSample, Configuration: x64-Release ------
  [1/5] Building CXX object CMakeFiles\ConanSample.dir\main.cpp.obj
  [2/5] Building CXX object CMakeFiles\ConanSample.dir\ConanSampleStateMachine.cpp.obj
  [3/5] Building CXX object CMakeFiles\ConanSample.dir\ConanSample.cpp.obj
  [4/5] Building CXX object CMakeFiles\ConanSample.dir\ConanSampleClass.cpp.obj
C:\Code\TangoConanSample\ConanSampleClass.cpp(227): warning C4101: 'end': unreferenced local variable
C:\Code\TangoConanSample\ConanSampleClass.cpp(227): warning C4101: 'start': unreferenced local variable
  [5/5] Linking CXX executable bin\ConanSample.exe
     Creating library lib\ConanSample.lib and object lib\ConanSample.exp

Build All succeeded.
```

Portability

- Want to test whether the Linux driver works nicely for your Windows DS?
  Use the same setup for multiple OS!
- Easily build the same DS for different flavors of Linux, e.g. support Debian, Ubuntu, Suse with the same code.
- Upgrade distros!

Image Attribution

- emojis designed by OpenMoji – the open-source emoji and icon project. License: CC BY-SA 4.0

Thank you for your attention!

Questions?