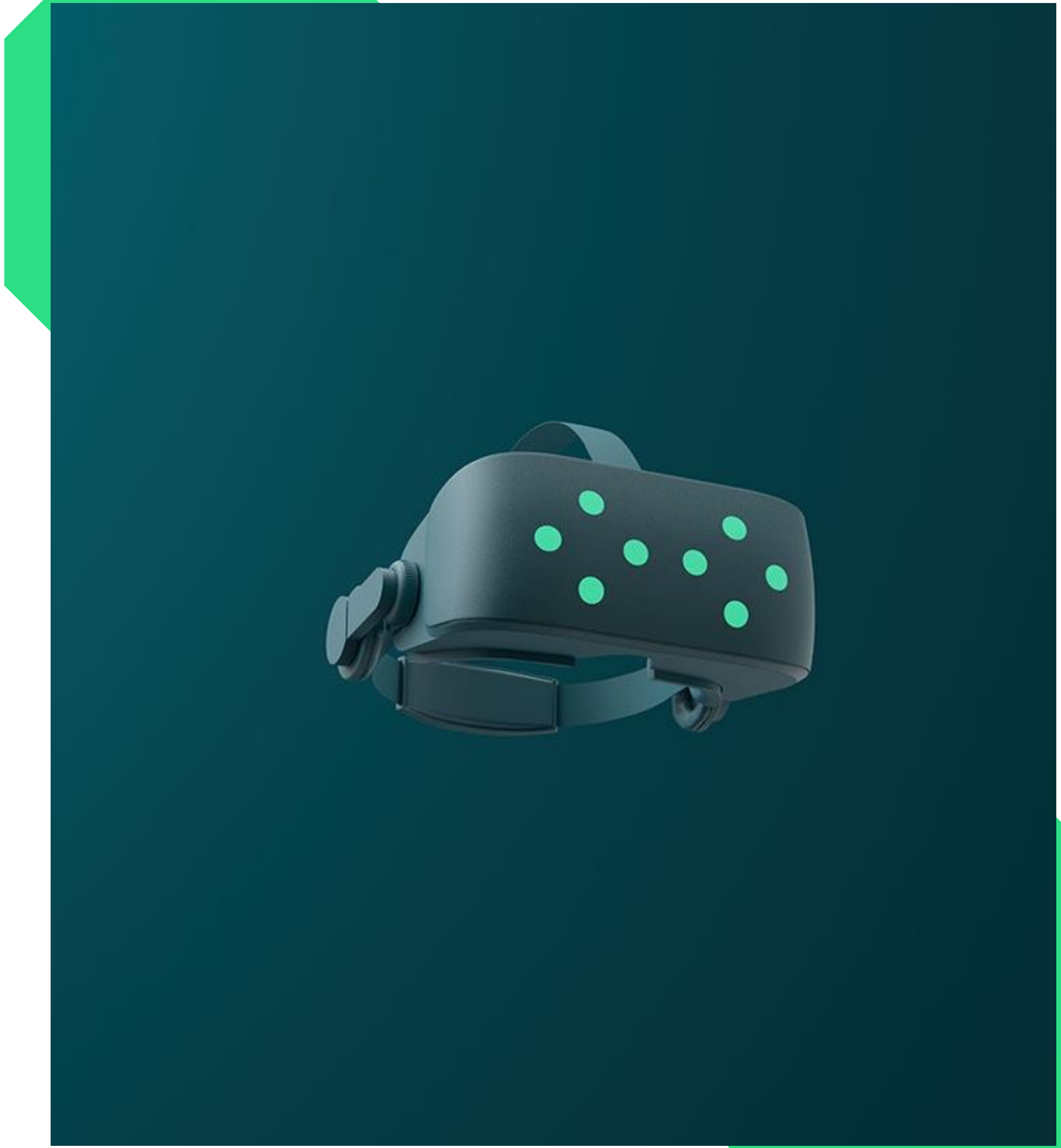
State and Direction of Qt's Build System

Alexandru Croitor

Cristian Le

02.05.2025

Qt Development



Agenda

1. What happened in Build System land
2. Research into improving public CMake API
3. Rust in Qt (what it means build system wise)

What happened?

1. Split of Public / Private modules into packages
2. Vcpkg integration
3. Qt Creator auto-installation of packages
4. Android Play Store Feature Delivery
5. Translation CMake API improvements
6. Qt Creator / Design Studio SBOM
7. Qt 5.15 SBOM
8. Modules.json schema
9. CI testing of unorthodox Qt builds
10. New Qt configure.cmake features
11. R&D of a better QML API

Split of Public / Private modules into packages

Contributed by Joerg Bornemann

- Starting with Qt 6.9 you need to call
  - find_package(Qt6 COMPONENTS CorePrivate)
  - This creates Qt6::CorePrivate which can be used to get access to private headers
- Better visibility of code which uses private API
- Allows cleaner split of files into distro-specific foo-devel packages

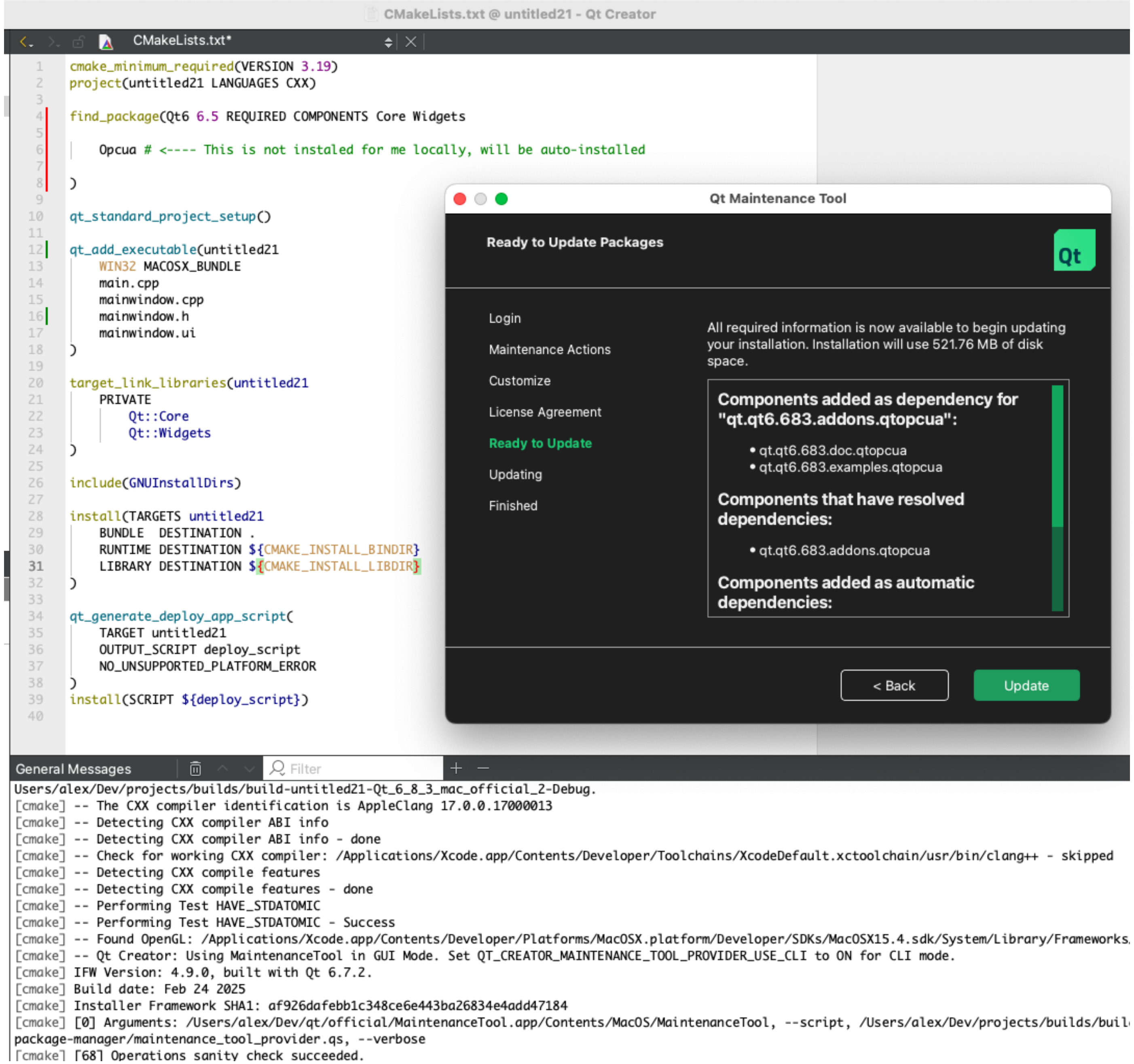
- Qt6Concurrent
- Qt6ConcurrentPrivate
- Qt6Core
- Qt6CorePrivate
- Qt6DBus
- Qt6DBusPrivate
- Qt6Designer
- Qt6DesignerPrivate
- Qt6Gui
- Qt6GuiPrivate
- Qt6Multimedia
- Qt6MultimediaPrivate

Qt Creator auto-installation of packages

Contributed by Cristian Adam

- Better integration of Qt Creator and CMake in regards to Qt package management
  - A project calls find_package(Qt6 REQUIRED COMPONENTS Opcua)
  - It will detect if the component is missing and will attempt to auto-install the component via the Qt Maintenance tool
  - Can be opted-out
  - Only enabled for Qt kits that are linked to Qt Creator via the Online installer / Maintenance tool

- Advantages
  - Faster acquisition of missing packages



```
cmake_minimum_required(VERSION 3.19)
project(untitled21 LANGUAGES CXX)

find_package(Qt6 6.5 REQUIRED COMPONENTS Core Widgets

    Opcua # <---- This is not instaled for me locally, will be auto-installed

)

qt_standard_project_setup()

qt_add_executable(untitled21
    WIN32 MACOSX_BUNDLE
    main.cpp
    mainwindow.cpp
    mainwindow.h
    mainwindow.ui
)

target_link_libraries(untitled21
    PRIVATE
        Qt::Core
        Qt::Widgets
)

include(GNUInstallDirs)

install(TARGETS untitled21
    BUNDLE  DESTINATION .
    RUNTIME DESTINATION ${CMAKE_INSTALL_BINDIR}
    LIBRARY DESTINATION ${CMAKE_INSTALL_LIBDIR}
)

qt_generate_deploy_app_script(
    TARGET untitled21
    OUTPUT_SCRIPT deploy_script
    NO_UNSUPPORTED_PLATFORM_ERROR
)
install(SCRIPT ${deploy_script})

```

All required information is now available to begin updating your installation. Installation will use 521.76 MB of disk space.

Components added as dependency for "qt.qt6.683.addons.qtopcua":

- qt.qt6.683.doc.qtopcua
- qt.qt6.683.examples.qtopcua

Components that have resolved dependencies:

- qt.qt6.683.addons.qtopcua

Components added as automatic dependencies:

```
Users/alex/Dev/projects/builds/build-untitled21-Qt_6_8_3_mac_official_2-Debug.
[cmake] -- The CXX compiler identification is AppleClang 17.0.0.17000013
[cmake] -- Detecting CXX compiler ABI info
[cmake] -- Detecting CXX compiler ABI info - done
[cmake] -- Check for working CXX compiler: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/clang++ - skipped
[cmake] -- Detecting CXX compile features
[cmake] -- Detecting CXX compile features - done
[cmake] -- Performing Test HAVE_STDATOMIC
[cmake] -- Performing Test HAVE_STDATOMIC - Success
[cmake] -- Found OpenGL: /Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX15.4.sdk/System/Library/Frameworks
[cmake] -- Qt Creator: Using MaintenanceTool in GUI Mode. Set QT_CREATOR_MAINTENANCE_TOOL_PROVIDER_USE_CLI to ON for CLI mode.
[cmake] IFW Version: 4.9.0, built with Qt 6.7.2.
[cmake] Build date: Feb 24 2025
[cmake] Installer Framework SHA1: af926dafebb1c348ce6e443ba26834e4add47184
[cmake] [0] Arguments: /Users/alex/Dev/qt/official/MaintenanceTool.app/Contents/MacOS/MaintenanceTool, --script, /Users/alex/Dev/projects/builds/buil
package-manager/maintenance_tool_provider.qs, --verbose
[cmake] [68] Operations sanity check succeeded.
```

Vcpkg integration

Contributed by Alexey Edelev

- Ongoing work on installing 3rd party dependencies in CI’s provisioning phase using vcpkg, rather than shell scripts
  - Currently limited to qtgrpc dependencies (protobuf)
  - Progress is slower than desired, due to CI flakiness, blocked integration and qt5.git turnaround

- Once merged, we’ll have the following benefits:
  - Easier acquiring of dependencies locally with same version as Qt CI
  - Avoids Powershell / Bash scripting
  - Paves the way for more 3rd party dependencies

Android Play Store Feature Delivery

Contributed by Alexey Edelev

- Google Play Feature Delivery allows certain features of an app to be delivered conditionally or downloaded on demand.
  - Projects need to separate these features from the base app into feature modules.
  - This will be handled by additional public CMake API

- Work in Progress, initial PoC might be available in Qt 6.10 or 6.11

Translation CMake API improvements

Contributed by Joerg Bornemann

- Various improvements to the Translations API
  - A way to merge Qt translations with project translations into a single file at build time
  - A better way to configure output location of translations
  - Various bug fixes

Qt Creator / Design Studio SBOM

Contributed by Alexandru Croitor

- Ongoing work to create and ship an SBOM for Qt Creator and Qt Design Studio
  - Based on the work done to create an SBOM for Qt
  - This will serve as inspiration for a future public SBOM API
- Currently Work In Progress
  - Depends on features in Qt 6.8.3+

Qt 5.15 SBOM

Contributed by Alexandru Croitor

- Ongoing work to create an SBOM for Qt 5.15
  - Less precise than the one in Qt 6.8
  - Is a mix of a source / analyze SBOM rather than a build SBOM
  - Works by scanning qmake .pro / .pri and qt_attribution.json files
  - Leverages our previous qmake2cmake tool

Modules.json schema

Contributed by Cristian Le

- Ongoing work to create a json schema for the Qt modules.json files
  - Core.json, Gui.json contain information about the ABI, Qt version, plugin types, compiler etc,
  - Useful for IDEs and tooling
  - The new schema will serve as a description on what the files contain

- Advantages:
  - Ensures better compatibility for the various IDEs and tools which consume the files
  - Allows testing and catching of regressions

- Currently a work in progress

CI testing of unorthodox Qt builds

Contributed by Alexandru Croitor

- Have you ever been annoyed that a top-level build of Qt fails to build?
  - or a no-prefix build?
  - or an in-source build?
  - or a combination of the above?
  - but it works fine in the CI…

- A new test suite is being developed to run all those configurations in a nigthly CI run
  - The test suite should catch these issues sooner than a few months after the breakage was merged
  - The suite also allows testing unmerged commits across a suite of repos by specifying which sha1s to check out
  - This could be improved the future to allow scheduling such builds directly via Gerrit or some bot
    - Would work as a Precheck on steroids for a set of unmerged changes across repos

New Qt configure.cmake features

Contributed by Cristian Le

- Better diagnosis of exclusive features like openssl vs openssl-linked
- Feature deprecation
- Presentation pending

R&D of a better QML API

Contributed by Cristian Le

- We should provide a nicer public CMake QML API to avoid option bloat of the current API
- Presentation pending

Thank you

alexandru.croitor@qt.io

cristian.le@qt.io

Qt Development