# This file controls Flutter-level build steps. It should not be edited.

cmake\_minimum\_required(VERSION 3.14)

set(EPHEMERAL\_DIR "${CMAKE\_CURRENT\_SOURCE\_DIR}/ephemeral")

# Configuration provided via flutter tool.

include(${EPHEMERAL\_DIR}/generated\_config.cmake)

# TODO: Move the rest of this into files in ephemeral. See

# https://github.com/flutter/flutter/issues/57146.

set(WRAPPER\_ROOT "${EPHEMERAL\_DIR}/cpp\_client\_wrapper")

# Set fallback configurations for older versions of the flutter tool.

if (NOT DEFINED FLUTTER\_TARGET\_PLATFORM)

set(FLUTTER\_TARGET\_PLATFORM "windows-x64")

endif()

# === Flutter Library ===

set(FLUTTER\_LIBRARY "${EPHEMERAL\_DIR}/flutter\_windows.dll")

# Published to parent scope for install step.

set(FLUTTER\_LIBRARY ${FLUTTER\_LIBRARY} PARENT\_SCOPE)

set(FLUTTER\_ICU\_DATA\_FILE "${EPHEMERAL\_DIR}/icudtl.dat" PARENT\_SCOPE)

set(PROJECT\_BUILD\_DIR "${PROJECT\_DIR}/build/" PARENT\_SCOPE)

set(AOT\_LIBRARY "${PROJECT\_DIR}/build/windows/app.so" PARENT\_SCOPE)

list(APPEND FLUTTER\_LIBRARY\_HEADERS

"flutter\_export.h"

"flutter\_windows.h"

"flutter\_messenger.h"

"flutter\_plugin\_registrar.h"

"flutter\_texture\_registrar.h"

)

list(TRANSFORM FLUTTER\_LIBRARY\_HEADERS PREPEND "${EPHEMERAL\_DIR}/")

add\_library(flutter INTERFACE)

target\_include\_directories(flutter INTERFACE

"${EPHEMERAL\_DIR}"

)

target\_link\_libraries(flutter INTERFACE "${FLUTTER\_LIBRARY}.lib")

add\_dependencies(flutter flutter\_assemble)

# === Wrapper ===

list(APPEND CPP\_WRAPPER\_SOURCES\_CORE

"core\_implementations.cc"

"standard\_codec.cc"

)

list(TRANSFORM CPP\_WRAPPER\_SOURCES\_CORE PREPEND "${WRAPPER\_ROOT}/")

list(APPEND CPP\_WRAPPER\_SOURCES\_PLUGIN

"plugin\_registrar.cc"

)

list(TRANSFORM CPP\_WRAPPER\_SOURCES\_PLUGIN PREPEND "${WRAPPER\_ROOT}/")

list(APPEND CPP\_WRAPPER\_SOURCES\_APP

"flutter\_engine.cc"

"flutter\_view\_controller.cc"

)

list(TRANSFORM CPP\_WRAPPER\_SOURCES\_APP PREPEND "${WRAPPER\_ROOT}/")

# Wrapper sources needed for a plugin.

add\_library(flutter\_wrapper\_plugin STATIC

${CPP\_WRAPPER\_SOURCES\_CORE}

${CPP\_WRAPPER\_SOURCES\_PLUGIN}

)

apply\_standard\_settings(flutter\_wrapper\_plugin)

set\_target\_properties(flutter\_wrapper\_plugin PROPERTIES

POSITION\_INDEPENDENT\_CODE ON)

set\_target\_properties(flutter\_wrapper\_plugin PROPERTIES

CXX\_VISIBILITY\_PRESET hidden)

target\_link\_libraries(flutter\_wrapper\_plugin PUBLIC flutter)

target\_include\_directories(flutter\_wrapper\_plugin PUBLIC

"${WRAPPER\_ROOT}/include"

)

add\_dependencies(flutter\_wrapper\_plugin flutter\_assemble)

# Wrapper sources needed for the runner.

add\_library(flutter\_wrapper\_app STATIC

${CPP\_WRAPPER\_SOURCES\_CORE}

${CPP\_WRAPPER\_SOURCES\_APP}

)

apply\_standard\_settings(flutter\_wrapper\_app)

target\_link\_libraries(flutter\_wrapper\_app PUBLIC flutter)

target\_include\_directories(flutter\_wrapper\_app PUBLIC

"${WRAPPER\_ROOT}/include"

)

add\_dependencies(flutter\_wrapper\_app flutter\_assemble)

# === Flutter tool backend ===

# \_phony\_ is a non-existent file to force this command to run every time,

# since currently there's no way to get a full input/output list from the

# flutter tool.

set(PHONY\_OUTPUT "${CMAKE\_CURRENT\_BINARY\_DIR}/\_phony\_")

set\_source\_files\_properties("${PHONY\_OUTPUT}" PROPERTIES SYMBOLIC TRUE)

add\_custom\_command(

OUTPUT ${FLUTTER\_LIBRARY} ${FLUTTER\_LIBRARY\_HEADERS}

${CPP\_WRAPPER\_SOURCES\_CORE} ${CPP\_WRAPPER\_SOURCES\_PLUGIN}

${CPP\_WRAPPER\_SOURCES\_APP}

${PHONY\_OUTPUT}

COMMAND ${CMAKE\_COMMAND} -E env

${FLUTTER\_TOOL\_ENVIRONMENT}

"${FLUTTER\_ROOT}/packages/flutter\_tools/bin/tool\_backend.bat"

${FLUTTER\_TARGET\_PLATFORM} $<CONFIG>

VERBATIM

)

add\_custom\_target(flutter\_assemble DEPENDS

"${FLUTTER\_LIBRARY}"

${FLUTTER\_LIBRARY\_HEADERS}

${CPP\_WRAPPER\_SOURCES\_CORE}

${CPP\_WRAPPER\_SOURCES\_PLUGIN}

${CPP\_WRAPPER\_SOURCES\_APP}

)