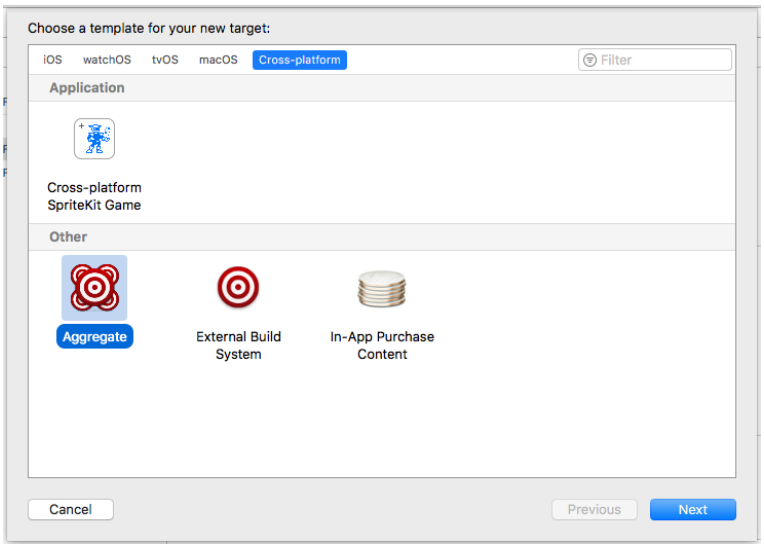
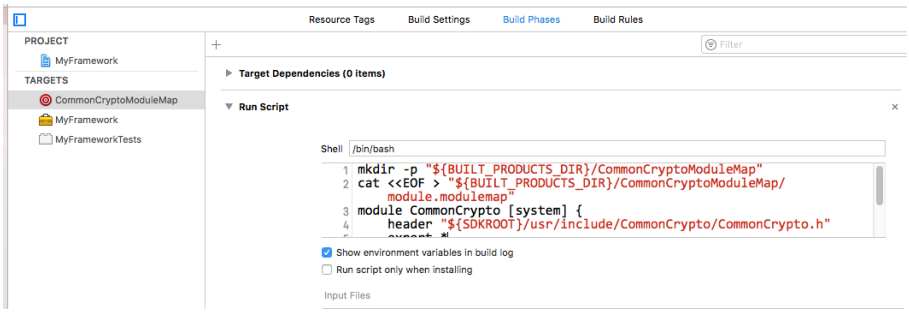
Something a little simpler and more robust is to create an Aggregate target called "CommonCryptoModuleMap" with a Run Script phase to generate the module map automatically and with the correct Xcode/SDK path:



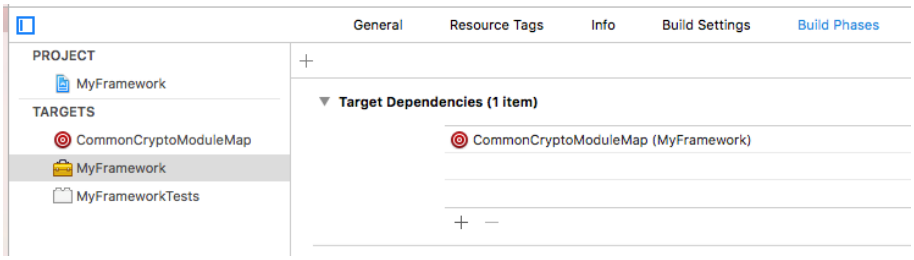


The Run Script phase should contain this bash:

```
# This if-statement means we'll only run the main script if the CommonCryptoModuleMap directory doesn't exist
# Because otherwise the rest of the script causes a full recompile for anything where CommonCrypto is a dependency
# Do a "Clean Build Folder" to remove this directory and trigger the rest of the script to run
if [ -d "${BUILT_PRODUCTS_DIR}/CommonCryptoModuleMap" ]; then
  echo "${BUILT_PRODUCTS_DIR}/CommonCryptoModuleMap directory already exists, so skipping the rest of the script."
  exit 0
fi

mkdir -p "${BUILT_PRODUCTS_DIR}/CommonCryptoModuleMap"
cat <<EOF > "${BUILT_PRODUCTS_DIR}/CommonCryptoModuleMap/module.modulemap"
module CommonCrypto [system] {
  header "${SDKROOT}/usr/include/CommonCrypto/CommonCrypto.h"
  export *
}
EOF
```

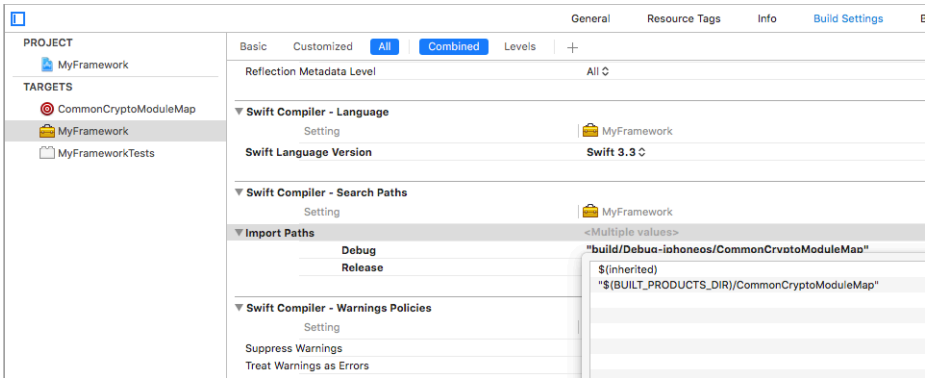
Using shell code and `${SDKROOT}` means you don't have to hard code the Xcode.app path which can vary system-to-system, especially if you use `xcode-select` to switch to a beta version, or are building on a CI server where multiple versions are installed in non-standard locations. You also don't need to hard code the SDK so this should work for iOS, macOS, etc. You also don't need to have anything sitting in your project's source directory.
After creating this target, make your library/framework depend on it with a Target Dependencies item:



This will ensure the module map is generated before your framework is built.

macOS note: If you're supporting `macOS` as well, you'll need to add `macosx` to the `Supported Platforms` build setting on the new aggregate target you just created, otherwise it won't put the module map in the correct `Debug` derived data folder with the rest of the framework products.

Next, add the module map's parent directory, `${BUILT_PRODUCTS_DIR}/CommonCryptoModuleMap`, to the "Import Paths" build setting under the Swift section (`SWIFT_INCLUDE_PATHS`):



Remember to add a `$(inherited)` line if you have search paths defined at the project or xcconfig level.

That's it, you should now be able to `import CommonCrypto`

answered Mar 17 '17
Mike Weller