Build Setting Reference

Product Information Build Settings

These build settings specify properties of the product the target builds.

ARCHS (Architectures)

Description:	Space-separated list of identifiers. Specifies the architectures (ABIs, processor models) to which the binary is targeted. When this build setting specifies more than one architecture, the generated binary may contain object code for each of the specified architectures.
Values:	See VALID_ARCHS (Valid Architectures).
Effector:	NATIVE_ARCH.
Default value:	`$NATIVE_ARCH`
Example value:	`ppc i386`
Companion:	VALID_ARCHS (Valid Architectures), ONLY_ACTIVE_ARCH (Build Active Architecture Only).
Prerequisite for:	PREBINDING (Prebinding).

DYLIB_COMPATIBILITY_VERSION (Compatibility Version)

Description:	Number. Specifies the compatibility version of a dynamic library product. See Dynamic Library Design Guidelines in *Dynamic Library Programming Topics* for details on assigning version numbers of dynamic libraries.
Default value:	`1`
Companion:	DYLIB_CURRENT_VERSION (Current Library Version).

DYLIB_CURRENT_VERSION (Current Library Version)

Description:	Number. Specifies the current version of a dynamic library product. See “Dynamic Library Design Guidelines” in *Dynamic Library Programming Topics* for details on assigning version numbers of dynamic libraries.
Default value:	`1`
Companions:	DYLIB_COMPATIBILITY_VERSION (Compatibility Version).

GENERATE_PKGINFO_FILE (Force Package Info Generation)

Description:	Boolean value. Specifies whether to generate the product’s package information file. For details on the package information file, see “Additional Configuration Tips” in *Runtime Configuration Guidelines*.

Values:	▪ `YES`: Generates the product's package information file. ▪ `NO`: Does not generate the product's package information file.
Default value:	▪ `YES`: In application targets. ▪ `NO`: In other target types.
Companions:	PKGINFO_FILE_PATH.

MACH_O_TYPE

Description:	Identifier. Specifies the binary's type. For information on binary types, see "Building Mach-O Files" in *Mach-O Programming Topics*.
Effector:	Target type, specified at the time the target is created.
Default value:	▪ `mh_executable`: Executable binary. Application, command-line tool, and kernel extension target types. ▪ `mh_bundle`: Bundle binary. Bundle and plug-in target types. ▪ `mh_object`: Relocatable object file. ▪ `mh_dylib`: Dynamic library binary. Dynamic library and framework target types. ▪ `staticlib`: Static library binary. Static library target types.
Affects:	GCC_ENABLE_SYMBOL_SEPARATION (Separate PCH Symbols), EXECUTABLE_EXTENSION.
Specified in:	New Project Assistant, New Target Assistant.

PRODUCT_NAME

Description:	Identifier. Specifies the name of the product the target builds.
Default value:	The name of the target at the time it was created.
Example value:	`MyProduct`
Affects:	EXECUTABLE_NAME, WRAPPER_NAME.

PROJECT_NAME

Description:	Identifier. Specifies the name of the project that defines the target.
Default value:	The name of the project at the time it was created.
Example value:	`MyProject`
Affects:	DSTROOT (Installation Build Products Location), PROJECT_TEMP_DIR.
Specified in:	Project navigator.

TARGET_NAME

Description:	Identifier. Identifies the target being processed.
Default value:	The name of the target at the time it was created.

Example value:	`MyProduct`
Affects:	TARGET_TEMP_DIR.
Specified in:	New Project Assistant.

VALID_ARCHS (Valid Architectures)

Description:	Space-separated list of identifiers. Specifies the architectures for which the binary may be built. During the build, this list is intersected with the value of `ARCHS` build setting; the resulting list specifies the architectures the binary can run on. If the resulting architecture list is empty, the target generates no binary.
Default value:	`m68k i386 sparc hppa ppc ppc7400 ppc970 ppc64 x86_64 armv6 armv7`
Affects:	CURRENT_ARCH.
Companion:	ARCHS (Architectures).

Build Properties Build Settings

These build settings specify properties of a build performed by a target.

ACTION

Description:	Identifier. Identifies the type of build to perform on the target.
Values:	▪ `build`: Build the product and place it in the product build directory (`CONFIGURATION_BUILD_DIR`). ▪ `clean`: Remove the product and build files in the product build directory (`CONFIGURATION_BUILD_DIR`) and the intermediate build files directory (`CONFIGURATION_TEMP_DIR`). ▪ `install`: Build the product and place it in its installation destination (`INSTALL_PATH`). ▪ `installhdrs`: Copy the product's public and private header files into the public headers directory (`PUBLIC_HEADERS_FOLDER_PATH`) and the private headers directory (`PRIVATE_HEADERS_FOLDER_PATH`), respectively. ▪ `installsrc`: Copy the target's source files into the project directory (`SRCROOT`).
Default value:	`build`: In `xcodebuild` invocations.
Affects:	BUILD_COMPONENTS, DEPLOYMENT_POSTPROCESSING (Deployment Postprocessing), DEPLOYMENT_LOCATION (Deployment Location).
Companions:	CONFIGURATION_BUILD_DIR (Per-Configuration Build Products Path), CONFIGURATION_TEMP_DIR (Per-Configuration Intermediate File Path), INSTALL_DIR, SRCROOT,PRIVATE_HEADERS_FOLDER_PATH,PUBLIC_HEADERS_FOLDER_PATH, INSTALLHDRS_COPY_PHASE.
Specified in:	▪ Xcode application: Product menu. ▪ `xcodebuild`: `<build_action>` argument.

BUILD_COMPONENTS

Description:	Space-separated list of identifiers. Specifies subsets of the product.
Effectors:	ACTION
	▪ `headers build`: When `$ACTION = build` or `$ACTION = install`,

Value:	• `headers`: When `$ACTION = installhdrs`, • *Empty*: When `$ACTION = installsrc`.

BUILD_VARIANTS (Build Variants)

Description:	Space-separated list of identifiers. Specifies the binary variants of the product. You can create additional variant names for special purposes. For example, you can use the name of a build configuration as a variant name to create highly customized binaries.
Values:	• `normal`: Use to produce a normal binary. • `profile`: Use to produce a binary that generates profile information. • `debug`: Use to produce a binary with debug symbols, additional assertions, and diagnostic code.
Default value:	`normal`
Affects:	CURRENT_VARIANT,OBJECT_FILE_DIR_<VARIANT>, OTHER_CFLAGS_<VARIANT>.

COMPRESS_PNG_FILES (Compress .png files)

Description:	Boolean value. Specifies whether to compress PNG files that are resources of the active target as they are copied to the application bundle. This applies only to iOS applications.
Values:	• `YES`: PNG files (those with the `.png` suffix) are compressed as they're copied to the application bundle. • `NO`: No PNG compression takes place.
Default value:	`YES`

CONFIGURATION

Description:	Identifier. Identifies the build configuration (for example, `Debug` or `Release`) the target uses to generate the product.
Values:	`Debug`, `Release`, and custom build configuration names.
Affects:	CURRENT_VARIANT, CONFIGURATION_BUILD_DIR (Per-Configuration Build Products Path), CONFIGURATION_TEMP_DIR (Per-Configuration Intermediate File Path).
Specified in:	• Schema > Build phase. • `xcodebuild -configuration`.

CURRENT_ARCH

Description:	Identifier. Identifies the architecture on which the build is being performed.
Values:	See ARCHS (Architectures).
Example value:	`i386`
Same as:	NATIVE_ARCH.

CURRENT_VARIANT

Description:	Identifier. Identifies the build variant being processed.
Effectors:	BUILD_VARIANTS (Build Variants), CONFIGURATION.
Value:	▪ `$CONFIGURATION`: When `$CONFIGURATION IN $BUILD_VARIANTS`. ▪ `normal`: Is the alternative.
Example values:	▪ `debug`: `$CONFIGURATION = debug AND $BUILD_VARIANTS = debug profile`. ▪ `normal`: `$CONFIGURATION = release AND $BUILD_VARIANTS = debug profile`.

DEBUG_INFORMATION_FORMAT (Debug Information Format)

Description:	Identifier. Identifies the format used to store the binary's debug information.
Values:	▪ `stabs`: Use the Stabs format and place the debug information in the binary. ▪ `dwarf`: Use the DWARF format and place the debug information in the binary. ▪ `dwarf-with-dsym`: Use the DWARF format and place the debug information in a dSYM file.
Default value:	`dwarf`
Prerequisite for:	GCC_ENABLE_SYMBOL_SEPARATION (Separate PCH Symbols).

DEPLOYMENT_POSTPROCESSING (Deployment Postprocessing)

Description:	Boolean value. Specifies whether the binary receives deployment postprocessing. Deployment postprocessing involves stripping the binary, and setting its file mode, owner, and group.
Effectors:	ACTION.
Values:	▪ `YES`: Binary receives deployment postprocessing. ▪ `NO`: Binary does not receive deployment postprocessing.
Default value:	▪ `YES`: When `$ACTION = install`. ▪ `NO`: Is the alternative.
Prerequisite for:	STRIP_INSTALLED_PRODUCT (Strip Linked Product).

ENABLE_HEADER_DEPENDENCIES

Description:	Boolean value. Specifies whether data gathered from header-file scans is used in the build process.
Values:	▪ `YES`: The build uses data gathered from header-file scans. ▪ `NO`: The build does not use data gathered from header-file scans.
Default value:	`YES`
Companion:	PATH_PREFIXES_EXCLUDED_FROM_HEADER_DEPENDENCIES.

NATIVE_ARCH

Description:	Identifier. Identifies the architecture on which the build is being performed (same as CURRENT_ARCH).
Values:	See ARCHS (Architectures).
Example value:	`i386`
Same as:	CURRENT_ARCH.
Companion:	ONLY_ACTIVE_ARCH (Build Active Architecture Only).

ONLY_ACTIVE_ARCH (Build Active Architecture Only)

Description:	Boolean value. Specifies whether the product includes only object code for the native architecture.
Values:	▪ `YES`: The product includes only code for the native architecture (NATIVE_ARCH). ▪ `NO`: The product includes code for the architectures specified in ARCHS (Architectures).
Default value:	`NO`

PATH_PREFIXES_EXCLUDED_FROM_HEADER_DEPENDENCIES

Description:	Space-separated list of directory paths. Identifies the directories to exclude from header-file scans when the build uses header-file dependencies.
Default value:	`/usr/include /usr/local/include /System/Library/Frameworks /System/Library/PrivateFrameworks /Developer/Headers`
Companions:	ENABLE_HEADER_DEPENDENCIES.

RETAIN_RAW_BINARIES

Description:	Boolean value. Specifies whether the binary is stripped.
Values:	▪ `YES`: Binary is not stripped. ▪ `NO`: Binary is stripped.
Default value:	`NO`
Affects:	BUILT_PRODUCTS_DIR.
Companion:	DEPLOYMENT_LOCATION (Deployment Location).
Related to:	SKIP_INSTALL.

STRINGS_FILE_OUTPUT_ENCODING

Description:	Identifier. Specifies the output encoding for strings files.
Values:	▪ `UTF-8` ▪ `UTF-16`
Default value:	`UTF-16`

TARGETED_DEVICE_FAMILY (Targeted Device Family)

Description:	Comma-separated list of numeric identifiers. Specifies the device families on which the product must be capable of running.
Identifiers:	▪ `1`: iPhone/iPod touch. ▪ `2`: iPad.
Default value:	`1`
Example value:	`1,2`

VALIDATE_PRODUCT (Validate Built Product)

Description:	Boolean value. Specifies whether to run product-validation tests.
Values:	▪ `YES`: The build runs validation tests on the generated product. ▪ `NO`: The build does not run validation tests on the generated product.
Default value:	`NO`

Build and Product Location Build Settings

These build settings identify filesystem locations used by the build process as well as locations that specify where product files are placed.

BUILT_PRODUCTS_DIR

Description:	Directory path. Identifies the directory under which all the product's files can be found. This directory contains either product files or symbolic links to them. Run Script build phases can use the value of this build setting as a convenient way to refer to the product files built by one or more targets even when these files are scattered throughout a directory hierarchy (for example, when `DEPLOYMENT_LOCATION` is set to `YES`.
Effectors:	RETAIN_RAW_BINARIES, CONFIGURATION_BUILD_DIR (Per-Configuration Build Products Path), DEPLOYMENT_LOCATION (Deployment Location).
Value:	▪ `$SYMROOT/BuiltProducts`: When `DEPLOYMENT_LOCATION = YES AND RETAIN_RAW_BINARIES = YES`, ▪ `$CONFIGURATION_BUILD_DIR`: Is the alternative.

CACHE_ROOT

	File path. Identifies the file used to cache build-time information that must persist between launches of the Xcode

Description:	application.
Value:	`/var/folders/<some_directory>/com.apple.Xcode.<user_id>`
Example value:	`/var/folders/Aq/AqPz2MexGfqyTWrWDAVsOE++12Q/-Caches-/com.apple.Xcode.501`
Affects:	SHARED_PRECOMPS_DIR (Precompiled Headers Cache Path).
Alias:	`CCHROOT`

CONFIGURATION_BUILD_DIR (Per-Configuration Build Products Path)

Description:	Directory path. Identifies the directory under which all build-related files for the active build configuration are placed.
Effectors:	CONFIGURATION,SYMROOT (Build Products Path).
Default value:	`$SYMROOT/$CONFIGURATION`
Example value:	`/Users/genica/MyProject/build/Debug`
Affects:	BUILT_PRODUCTS_DIR, TARGET_BUILD_DIR, TARGET_TEMP_DIR.

CONFIGURATION_TEMP_DIR (Per-Configuration Intermediate File Path)

Description:	Directory path. Identifies the directory that holds temporary files for the active build configuration.
Effectors:	CONFIGURATION, PROJECT_TEMP_DIR.
Default value:	`$PROJECT_TEMP_DIR/$CONFIGURATION`
Example value:	`/Users/genica/MyProject/build/MyProject.build/Debug`
Affects:	TARGET_TEMP_DIR.

DEPLOYMENT_LOCATION (Deployment Location)

Description:	Boolean value. Specifies whether product files are placed in the installation (specified by `DSTROOT`) or the build directory (identified by `SYMROOT`).
Effector:	ACTION.
Values:	▪ `YES`: Product files are placed in `$DSTROOT`. ▪ `NO`: Product files are placed in `$SYMROOT`.
Default value:	▪ `YES`: When `$ACTION = install`. ▪ `NO`: Is the alternative.
Affects:	TARGET_BUILD_DIR.
Companions:	DSTROOT (Installation Build Products Location), SYMROOT (Build Products Path),.
Related to:	RETAIN_RAW_BINARIES,BUILT_PRODUCTS_DIR, SKIP_INSTALL.

DERIVED_FILE_DIR

Description:	Directory path. Identifies the directory into which derived source files—such as those generated by lex and yacc—are placed.
Effectors:	TARGET_TEMP_DIR.
Value:	`$TARGET_TEMP_DIR/DerivedSources`
Aliases:	`DERIVED_FILES_DIR`, `DERIVED_SOURCES_DIR`

DSTROOT (Installation Build Products Location)

Description:	Directory path. Identifies the directory into which the product is placed. In this directory, the product is laid out exactly as it would be installed in a user's filesystem.
Effectors:	PROJECT_NAME.
Default value:	`/tmp/$PROJECT_NAME.dst`
Example value:	`/tmp/MyProject.dst`
Affects:	INSTALL_DIR, TARGET_BUILD_DIR.

INSTALL_DIR

Description:	Directory path. Identifies the directory in the developer's filesystem into which the *installed* product is placed.
Effectors:	DSTROOT (Installation Build Products Location), INSTALL_PATH (Installation Directory).
Value:	`$DSTROOT/INSTALL_PATH`
Example value:	`/tmp/MyProduct.dst/Users/genica/Library/Bundles`

INSTALL_PATH (Installation Directory)

Description:	Directory path. Identifies the directory in the user's filesystem into which the installed product is placed.
Effectors:	Product type (chosen when the project was created), DSTROOT (Installation Build Products Location), SYSTEM_LIBRARY_DIR, USER_LIBRARY_DIR, HOME.
Default value:	`$SYSTEM_LIBRARY_DIR/Extensions`: Kernel extension project. `$USER_LIBRARY_DIR/Automator`: Action project. `$HOME/Applications`: Application project. `$HOME/Library/Bundles`: Audio unit and bundle projects. `$HOME/bin`: Command-line utility project. `$DSTROOT`: Apple plug-in project (complete path depends on specific project template). `/usr/local/lib`: Dynamic library and static library projects.
Affects:	INSTALL_DIR, TARGET_BUILD_DIR.

OBJECT_FILE_DIR

Description:	Directory path. Partially identifies the directory into which variant object files are placed. The complete specification is computed using the variants of this build setting.
Effectors:	TARGET_TEMP_DIR.
Value:	`$TARGET_TEMP_DIR/Objects`
Example value:	`/Volumes/Users/genica/MyProject/build/MyProject.build/Debug/MyProduct.build/Objects`
Affects:	OBJECT_FILE_DIR_<VARIANT>.

OBJECT_FILE_DIR_<VARIANT>

Description:	Directory path. Fully identifies the directory into which variant object files are placed. For each build variant in `BUILD_VARIANTS`, Xcode generates an `OBJECT_FILE_DIR` build setting with the variant name as a suffix. The generated build setting's value is computed using `OBJECT_FILE_DIR` and the build variant name.
Effectors:	BUILD_VARIANTS (Build Variants), OBJECT_FILE_DIR.
Value:	`$OBJECT_FILE_DIR-<VARIANT>`
Example build settings and their values when `$BUILD_VARIANTS = normal debug`:	▪ `$OBJECT_FILE_DIR_normal = /Volumes/Users/genica/MyProject/build/MyProject.build/Debug/MyProduct.build/Objects-normal` ▪ `$OBJECT_FILE_DIR_debug = /Volumes/Users/genica/MyProject/build/MyProject.build/Debug/MyProduct.build/Objects-debug`
Related to:	BUILD_VARIANTS (Build Variants), OTHER_CFLAGS_<VARIANT>.

OBJROOT (Intermediate Build Files Path)

Description:	Directory path. Identifies the directory in which the target's intermediate build files are placed. Intermediate build directories are named after the product name with the extension `.build`. For example, `MyProduct.build`.
Effectors:	SRCROOT, Xcode Preferences > Building > "Place Build Products in."
Default value:	▪ `$SRCROOT/build`: When Xcode Preferences > Building > "Place Build Products in" is "Project directory." ▪ `<custom_directory_path>`: When Xcode Preferences > Building > "Place Build Products in" is "Customized location."
Example value:	`/Volumes/Users/genica/MyProject/build`
Affects:	PROJECT_TEMP_DIR.

PROJECT_TEMP_DIR

Description:	Directory path. Identifies the directory in which the project's intermediate build files are placed. This directory is shared between all the targets defined by the project. Run Script build phases should generate intermediate build files in the directory identified by `DERIVED_FILE_DIR`, not the location this build setting specifies.
Effectors:	PROJECT_NAME,OBJROOT (Intermediate Build Files Path).
Value:	`$OBJROOT/$PROJECT_NAME.build`
Example value:	`/Volumes/Users/genica/MyProject/build/MyProject.build`
Affects:	

	CONFIGURATION_TEMP_DIR (Per-Configuration Intermediate File Path).

REZ_COLLECTOR_DIR

Description:	Directory path. Specifies the directory in which the collected Resource Manager resources generated by `ResMerger` are stored before they are added to the product.
Effectors:	TARGET_TEMP_DIR.
Value:	`$TARGET_TEMP_DIR/ResourceManagerResources`
Example value:	`/Volumes/Users/genica/MyProject/build/MyProject.build/Debug/MyProduct.build/ResourceManagerResources`
Affects:	REZ_OBJECTS_DIR.

REZ_OBJECTS_DIR

Description:	Directory path. Specifies the directory in which compiled Resource Manager resources generated by `Rez` are stored before they are collected using ResMerger.
Effectors:	REZ_COLLECTOR_DIR.
Value:	`$REZ_COLLECTOR_DIR/Objects`
Example value:	`/Volumes/Users/genica/MyProject/build/MyProject.build/Debug/MyProduct.build/ResourceManagerResources/Object`

SDKROOT (Base SDK)

Description:	Directory path. Specifies the directory of the base SDK to use to build the product.
Values:	▪ `macosx10.5`: OS X v10.5. ▪ `macosx10.6`: OS X v10.6. ▪ `iphonesimulator3.2`: iPhone Simulator 3.2. ▪ `iphonesimulator4.0`: iPhone Simulator 4.0. ▪ `iphoneos3.2`: iPhone Device 3.2. ▪ `iphoneos4.0`: iPhone Device 4.0.
Related to:	FRAMEWORK_SEARCH_PATHS (Framework Search Paths), HEADER_SEARCH_PATHS (Header Search Paths), IPHONEOS_DEPLOYMENT_TARGET (iOS Deployment Target), MACOSX_DEPLOYMENT_TARGET (OS X Deployment Target).

SHARED_PRECOMPS_DIR (Precompiled Headers Cache Path)

Description:	Directory path. Specifies the directory in which to place precompiled headers. Targets can share precompiled headers by specifying the same value for this build setting.
Effectors:	CACHE_ROOT.
Default value:	`$CACHE_ROOT/SharedPrecompiledHeaders`
Example value:	`/var/folders/Aq/AqPz2MexGfqyTWrWDAVsOE++12Q/-Caches-/com.apple.Xcode.501/SharedPrecompiledHeaders`

SKIP_INSTALL

Description:	Boolean value. Specifies whether to place the product at the location indicated by `DSTROOT` or the uninstalled products directory inside the directory indicated by `TARGET_TEMP_DIR`.
Values:	▪ `YES`: When `$DEPLOYMENT_LOCATION = YES`, the product is placed in `$TARGET_TEMP_DIR/UninstalledProducts`. ▪ `NO`: The product is placed in `$DSTROOT`.
Default value:	`NO`
Affects:	TARGET_BUILD_DIR.
Companions:	DEPLOYMENT_LOCATION (Deployment Location), DSTROOT (Installation Build Products Location), TARGET_TEMP_DIR.

SRCROOT

Description:	Directory path. Identifies the directory containing the target's source files.
Value:	Path to the project file that defines the target.
Example value:	`/Volumes/Users/genica/MyProject`
Affects:	OBJROOT (Intermediate Build Files Path), SYMROOT (Build Products Path).
Alias:	`SOURCE_ROOT`

SYMROOT (Build Products Path)

Description:	Directory path. Identifies the root of the directory hierarchy that contains product files and intermediate build files. Product and build files are placed in subdirectories of this directory.
Effectors:	SRCROOT, Xcode Preferences > Build.
Default value:	▪ `$SRCROOT/build`: When Xcode Preferences > Build > "Place Build Products in" is "Project Directory." ▪ `<custom_directory_path>`: When Xcode Preferences > Build > "Place Build Products in" is "Custom location."
Example values:	▪ `/Volumes/Users/genica/MyProject/build` ▪ `/Volumes/A_Volume/MyManyProducts`
Affects:	BUILT_PRODUCTS_DIR, CONFIGURATION_BUILD_DIR (Per-Configuration Build Products Path).

TARGET_BUILD_DIR

Description:	Directory path. Identifies the root of the directory hierarchy that contains the product's files (no intermediate build files). Run Script build phases that operate on product files of the target that defines them should use the value of this build setting. But Run Script build phases that operate on product files of other targets should use BUILT_PRODUCTS_DIR instead.
Effectors:	CONFIGURATION_BUILD_DIR (Per-Configuration Build Products Path), DEPLOYMENT_LOCATION (Deployment Location), DSTROOT (Installation Build Products Location), INSTALL_PATH (Installation Directory), TARGET_TEMP_DIR, SKIP_INSTALL.

Value:	▪ $CONFIGURATION_BUILD_DIR: When $DEPLOYMENT_LOCATION = NO, ▪ $DSTROOT/$INSTALL_PATH: When $DEPLOYMENT_LOCATION = YES, $SKIP_INSTALL = NO, and INSTALL_PATH is defined, ▪ $TARGET_TEMP_DIR/UninstalledProducts: When $DEPLOYMENT_LOCATION = YES AND $SKIP_INSTALL = YES or $SKIP_INSTALL = NO and INSTALL_PATH is not defined.
Example values:	▪ /Volumes/Users/genica/MyProject/build/Debug ▪ /tmp/MyProject.dst/Users/genica/Applications ▪ /Volumes/Users/genica/MyProject/build/UninstalledProducts
Related to:	DEPLOYMENT_LOCATION (Deployment Location), INSTALL_PATH (Installation Directory), SKIP_INSTALL.

TARGET_TEMP_DIR

Description:	Directory path. Identifies the directory containing the target's intermediate build files. Run Script build phases should place intermediate files at the location indicated by DERIVED_FILE_DIR, not the directory identified by this build setting.
Effectors:	TARGET_NAME, CONFIGURATION_TEMP_DIR (Per-Configuration Intermediate File Path).
Value:	$CONFIGURATION_TEMP_DIR/$TARGET_NAME.build
Example value:	/Volumes/Users/genica/MyProject/build/MyProject.build/Debug/MyProduct.build
Affects:	DERIVED_FILE_DIR, OBJECT_FILE_DIR, REZ_COLLECTOR_DIR, TARGET_BUILD_DIR.

Header-Map Build Settings

Header maps (also known as "header maps") are files Xcode uses to compile the locations of the headers used in a target. These files use the suffix .hmap. Xcode passes the header maps it puts together to C-based compilers through the -I argument.

They allow header and source files to include:

- Any header associated with the target using only its name (for example, #include "MyClass.h") regardless of its location in the file system. See HEADERMAP_INCLUDES_FLAT_ENTRIES_FOR_TARGET_BEING_BUILT.
- Headers using framework syntax (for example, MyFramework/MyHeader.h). See HEADERMAP_INCLUDES_FRAMEWORK_ENTRIES_FOR_ALL_PRODUCT_TYPES.
- Headers that are part of the project, regardless or target membership. See HEADERMAP_INCLUDES_PROJECT_HEADERS.

Without header maps, you need to add each directory that contains headers to the target's header search paths (see HEADER_SEARCH_PATHS (Header Search Paths) and USER_HEADER_SEARCH_PATHS (User Header Search Paths)).

HEADERMAP_INCLUDES_FLAT_ENTRIES_FOR_TARGET_BEING_BUILT

Description:	Boolean value. Specifies whether the header map contains a name/path entry for every header in the target being built.
Values:	▪ YES: The header map contains a name/path entry for every header in the target. ▪ NO: The header map does not contain name/path entries for the headers that belong to the target.
Default value:	YES
Related to:	HEADERMAP_INCLUDES_FRAMEWORK_ENTRIES_FOR_ALL_PRODUCT_TYPES,HEADERMAP_INCLUDES_PROJECT_HEADERS.

HEADERMAP_INCLUDES_FRAMEWORK_ENTRIES_FOR_ALL_PRODUCT_TYPES

Description:	Boolean value. Specifies whether the header map contains a framework-name/path entry for every header in the target being built, including targets that do not build frameworks.
Values:	▪ YES: The header map contains a framework-name/path entry for every header in the target. ▪ NO: The header map does not contain framework-name/path entries for the headers in the target.
Default value:	YES
Related to:	HEADERMAP_INCLUDES_FLAT_ENTRIES_FOR_TARGET_BEING_BUILT, HEADERMAP_INCLUDES_PROJECT_HEADERS.

HEADERMAP_INCLUDES_PROJECT_HEADERS

Description:	Boolean value. Specifies whether the header map contains a name/path entry for every header in the project, regardless of the headers' target membership.
Values:	▪ YES: The header map contains a name/path entry for every header in the project. ▪ NO: The header map does not contain name/path entries for the headers that are part of the project.
Default value:	YES
Related to:	HEADERMAP_INCLUDES_FLAT_ENTRIES_FOR_TARGET_BEING_BUILT, HEADERMAP_INCLUDES_FRAMEWORK_ENTRIES_FOR_ALL_PRODUCT_TYPES.

C/C++ Compiler Build Settings

These build settings specify how source files are compiled into object files.

ALWAYS_SEARCH_USER_PATHS (Always Search User Paths)

Description:	Boolean value. Specifies whether the compiler searches for headers in the project directory before searching system directories. This build setting is used only with GCC 4.0 and later.
Values:	▪ YES: Search project directory first. ▪ NO: Search system directories first.
Default value:	YES. For backwards compatibility only. You should set this build setting to NO.

FRAMEWORK_SEARCH_PATHS (Framework Search Paths)

Description:	Space-separated list of directory paths. Specifies directories in which the compiler searches for frameworks to find included header files. This list is passed to the compiler in the gcc -F option. You may specify a recursive path by appending ** to the path. When this build setting is defined, $SDKROOT is added to the end of the path list that is passed to the compiler.
Default value:	None.
Example values:	▪ /Users/genica/TestFrameworks/** ▪ /Volumes/Auryon/TeamFrameworks/**
Companions:	SDKROOT (Base SDK).

GCC_AUTO_VECTORIZATION (Auto-Vectorization)

Description:	Boolean value. Specifies whether the compiler performs automatic loop vectorization when appropriate. Automatic loop vectorization is supported only in the PPC architectures. And it's not supported by the Clang and LLVM-GCC compilers.
Prerequisite:	`$GCC_OPTIMIZATION_LEVEL >= 2 AND $ARCHS * $VALID_ARCHS IN {ppc, ppc970, ppc64}`
Values:	▪ `YES`: The compiler performs automatic loop vectorization when the prerequisite is met. ▪ `NO`: The compiler does not perform automatic loop vectorization.
Default value:	`NO`
Companions:	ARCHS (Architectures), VALID_ARCHS (Valid Architectures), GCC_OPTIMIZATION_LEVEL (Optimization Level).

GCC_CW_ASM_SYNTAX (CodeWarrior-Style Inline Assembly)

Description:	Boolean value. Specifies whether to use the CodeWarrior syntax for inline assembly code (in addition to the standard GCC syntax).
Values:	▪ `YES`: Use CodeWarrior syntax for inline assembly code. ▪ `NO`: Do not use CodeWarrior syntax for inline assembly code.
Default value:	`YES`

GCC_DEBUGGING_SYMBOLS (Level of Debug Symbols)

Description:	Option specification. Specifies the level of debug information included in the binary.
Values:	`used`: Referenced symbols only (`gcc -gused`). `full`: All symbols (`gcc -gfull`). `default`: Compiler default (`gcc -g`).
Default value:	`default`
Prerequisite for:	GCC_ENABLE_SYMBOL_SEPARATION (Separate PCH Symbols), DEAD_CODE_STRIPPING (Dead Code Stripping)

GCC_DYNAMIC_NO_PIC

Description:	Boolean value. Specifies whether the generated object code is nonrelocatable (external references remain relocatable). Making code nonrelocatable results in faster function calls. This feature is appropriate in applications but not dynamic libraries.
Values:	▪ `YES`: Generated code is nonrelocatable (`gcc -mdynamic-no-pic`) when the prerequisite is met. ▪ `NO`: Generated code is relocatable.
Default value:	`NO`

GCC_ENABLE_CPP_EXCEPTIONS (Enable C++ Exceptions)

Description:	Boolean value. Specifies whether the compiler generates code necessary for exception propagation.
Values:	▪ `YES`: Compiler generates code necessary for exception propagation. ▪ `NO`: Compiler does not generate code necessary for exception propagation.
Default value:	`NO`
Related to:	GCC_ENABLE_CPP_RTTI (Enable C++ Runtime Types).

GCC_ENABLE_CPP_RTTI (Enable C++ Runtime Types)

Description:	Boolean value. Specifies whether the compiler generates information about every class with virtual functions. This information is used by the C++ runtime type identification features (`dynamic_cast` and `typeid`). If you do not use these features, you may save some space by not generating this information. However, when exceptions are enabled, this information is generated automatically.
Values:	▪ `YES`: Binary includes information about virtual classes. ▪ `NO`: Binary might not include information about virtual classes (`gcc -fno-rtti`).
Default value:	`YES`
Related to:	GCC_ENABLE_CPP_EXCEPTIONS (Enable C++ Exceptions).

GCC_ENABLE_FIX_AND_CONTINUE (Fix & Continue)

Description:	Boolean value. Specifies whether the binary uses Fix And Continue..
Values:	▪ `YES`: Binary uses Fix And Continue. ▪ `NO`: Binary does not use Fix And Continue.
Default value:	`NO`

GCC_ENABLE_OBJC_EXCEPTIONS (Enable Objective-C Exceptions)

Description:	Boolean value. Specifies whether the compiler recognizes `@try`, `@catch`, and `@throw` directives.
Values:	▪ `YES`: Recognize the Objective-C exception-handling directives (`gcc -fobjc-exceptions`). ▪ `NO`: Do not allow the Objective-C exception-handling directives in source code.
Default value:	`NO`

GCC_ENABLE_OBJC_GC (Objective-C Garbage Collection)

Description:	Identifier. Specifies the level of garbage-collection support for the generated code.
	▪ `unsupported`: The application cannot load code that requires garbage collection. The loadable bundle cannot be loaded by an application that requires garbage collection.

Values:	▪ `supported`: The application can load code that supports or requires garbage collection. The loadable bundle can be loaded by an application with any level of garbage-collection support. ▪ `required`: The application can load only code that supports garbage collection. The loadable bundle can be loaded only by an application that supports garbage collection.
Default value:	`unsupported`

GCC_ENABLE_SSE3_EXTENSIONS (Enable SSE3 Extensions)

Description:	Boolean value. Specifies whether the binary uses the built-in functions that provide access to the SSE3 extensions to the IA-32 architecture.
Values:	▪ `YES`: Binary uses SSE3 functions. ▪ `NO`: Binary does not use SSE3 functions.
Default value:	`NO`

GCC_ENABLE_SSE41_EXTENSIONS (Enable SSE4.1 Extensions)

Description:	Boolean value. Specifies whether the binary uses the built-in functions that provide access to the SSE4.1 extensions to the IA-32 architecture.
Values:	▪ `YES`: Binary uses SSE4.1 functions (`gcc -msse4.1`). ▪ `NO`: Binary does not use SSE4.1 functions.
Default value:	`NO`

GCC_ENABLE_SSE42_EXTENSIONS (Enable SSE4.2 Extensions)

Description:	Boolean value. Specifies whether the binary uses the built-in functions that provide access to the SSE4.2 extensions to the IA-32 architecture.
Values:	▪ `YES`: Binary uses SSE4.2 functions (`gcc -msse4.2`). ▪ `NO`: Binary does not use SSE4.2 functions.
Default value:	`NO`

GCC_ENABLE_SYMBOL_SEPARATION (Separate PCH Symbols)

Description:	Boolean value. Specifies whether the compiler generates a separate file containing the debug symbols when compiling a precompiled (prefix) header (PCH). A separate file with debug symbols can improve build time.
Prerequisite:	`$DEBUG_INFORMATION_FORMAT = stabs AND $GCC_DEBUGGING_SYMBOLS = full`
Values:	▪ `YES`: Generates separate file containing debug symbols for a precompiled header. ▪ `NO`: Does not generate separate debug symbol file.
Default	▪ `YES`: When `$MACH_O_TYPE != staticlib`.

value:	▪ NO: Is the alternative.
Effector:	MACH_O_TYPE
Companions:	DEBUG_INFORMATION_FORMAT (Debug Information Format), GCC_DEBUGGING_SYMBOLS (Level of Debug Symbols).

GCC_FEEDBACK_DIRECTED_OPTIMIZATION (Feedback-Directed Optimization)

Description:	Boolean value. Specifies whether to use feedback-directed optimization. To optimize a binary, you must first generate a binary that produces profile trace files by setting this build setting to GenerateProfile. After running the binary mimicking the expected usage patterns (training), rebuild the binary with UseProfile as the value for this build setting. The resulting binary is optimized for the usage patterns observed in training. If the code paths taken during training are not representative of what happens in actual usage, the binary's performance may actually degrade.
Values:	▪ Off: Binary is not optimized and does not generate trace files. ▪ GenerateProfile: Binary generates trace files (training). ▪ UseProfile: Binary is optimized using the information from the profile trace files. Requires that the binary had been previously built with GenerateProfile and run to gather the information.
Default value:	Off

GCC_GENERATE_DEBUGGING_SYMBOLS (Generate Debug Symbols)

Description:	Boolean value. Specifies whether the binary includes debug symbols.
Values:	▪ YES: Binary includes debugging symbols. ▪ NO: Binary does not include debugging symbols.
Default value:	YES
Related to:	GCC_DEBUGGING_SYMBOLS (Level of Debug Symbols).

GCC_MODEL_TUNING (Instruction Scheduling)

Description:	Option specification. Specifies the PowerPC architecture to which the compiler optimizes the instruction scheduling model. The generated code runs in earlier PowerPC architectures, too. See -mtune in the gcc man page for details.
Values:	▪ None: Binary is not optimized for a particular PowerPC architecture. ▪ G3: Binary is optimized for the PowerPC G3 architecture. ▪ G4: Binary is optimized for the PowerPC G4 architecture. ▪ G5: Binary is optimized for the PowerPC G5 architecture.
Default value:	G4

GCC_OBJC_CALL_CXX_CDTORS (Call C++ Default Ctors/Dtors in Objective-C)

Description:	Boolean value. Specifies whether to execute nontrivial default constructors and destructors for C++ instance variables of Objective-C classes.

Values:	• YES: Binary executes default constructors and destructors for C++ instance variables of Objective-C classes (gcc -fobjc-call-cxx-cdtors). • NO: Binary does not execute default constructors for Objective-C-typed instance variables in C++ classes.
Default value:	NO

GCC_OPTIMIZATION_LEVEL (Optimization Level)

Description:	Option specification. Specifies the degree to which the generated code is optimized for speed and binary size.
Values:	• 0: No optimization. • 1: Binary is optimized to *fast*. • 2: Binary is optimized to *faster*. • 3: Binary is optimized to *fastest*. • s: Binary is optimized to *fastest and smallest*.
Default value:	s

GCC_PRECOMPILE_PREFIX_HEADER (Precompile Prefix Header)

Description:	Boolean value. Specifies whether to create a prefix header for the target.
Prerequisite:	$GCC_PREFIX_HEADER identifies an existing prefix header.
Values:	• YES: Target generates a prefix header when the prerequisite is met. • NO: Target does not generate a prefix header.
Default value:	NO
Companion:	GCC_PREFIX_HEADER.

GCC_PREFIX_HEADER

Description:	Filename or file path. Identifies the target's prefix header.
Default value:	None.
Example value:	MyProduct_Prefix.pch
Prerequisite for:	GCC_PRECOMPILE_PREFIX_HEADER (Precompile Prefix Header)

GCC_PREPROCESSOR_DEFINITIONS (Preprocessor Macros)

Description:	Space-separated list of option specifications. Specifies preprocessor macros in the form foo (for a simple #define) or foo=1 (for a value definition). This list is passed to the compiler through the gcc -D option when compiling precompiled headers and implementation files.
Default value:	None.
Example value:	test_mode=1 copious_logging=1

Related to:	GCC_PREPROCESSOR_DEFINITIONS_NOT_USED_IN_PRECOMPS (Preprocessor Macros Not Used In Precompiled Headers).

GCC_PREPROCESSOR_DEFINITIONS_NOT_USED_IN_PRECOMPS (Preprocessor Macros Not Used In Precompiled Headers)

Description:	Space-separated list of option specifications. Specifies preprocessor macros in the form `foo` (for a simple `#define`) or `foo=1` (for a value definition). This list is passed to the compiler through the `gcc -D` option only when compiling implementation files; they are not passed when compiling precompiled headers.
Prerequisite:	Definitions used only in implementation files, not precompiled headers.
Default value:	None.
Example value:	`test_mode=1 copious_logging=1`
Related to:	GCC_PREPROCESSOR_DEFINITIONS (Preprocessor Macros).

GCC_SYMBOLS_PRIVATE_EXTERN (Symbols Hidden by Default)

Description:	Boolean value. Specifies whether symbols are hidden by default. See Controlling Symbol Visibility in *C++ Runtime Environment Programming Guide*.
Values:	▪ `YES`: Symbols that do not specify public visibility (with `__attribute__((visibility("default")))`, for example) are not exported (`gcc -fvisibility=hidden`). ▪ `NO`: Symbols that do not specify private visibility (with `__attribute__((visibility("hidden")))`, for example) are exported.
Default value:	`YES`
Prerequisite for:	STANDARD_C_PLUS_PLUS_LIBRARY_TYPE (C++ Standard Library Type).

GCC_THREADSAFE_STATICS (Statics are Thread Safe)

Description:	Boolean value. Specifies whether the binary uses the functions that implement thread-safe initialization of local statics for the IA-32 architecture. Binaries that use these functions contain less object code in sections that do not need to be thread safe.
Values:	▪ `YES`: Binary uses the IA-32 ABI thread-safe initialization functions. ▪ `NO`: Binary does not use the IA-32 ABI thread-safe initialization functions (`gcc -fno-threadsafe-statics`).
Default value:	`YES`

GCC_UNROLL_LOOPS (Unroll Loops)

Description:	Boolean value. Specifies whether the compiler generates a faster binary (containing code with fewer branches) by unrolling loops, which generates a larger binary.
Values:	▪ `YES`: Compiler generates code with unrolled loops.

	• NO: Compiler does not unroll loops.
Default value:	NO

GCC_USE_NASM_FOR_ASM_FILETYPE (Use nasm to Process .asm Files)

Description:	Boolean value. Specifies whether nasm is used to compile Assembly .asm files.
Values:	• YES: Assembly (.asm) files are compiled with nasm (gcc -nasm). • NO: Assembly files are not compiled with nasm.
Default value:	NO

GCC_VERSION

Description:	Numeric identifier. Identifies the GCC version to be used to compile the target's source files. When the target's "System C rule" is set to GCC System Version (instead of a specific version number), this build setting is not available in Run Script build phases.
Values:	• 2.95.2 • 3.1 • 3.3 • 4.0
Default value:	GCC system version.
Specified in:	• Project Info > Rules > "System C rule." • Target Info > Rules > "System C rule."
Affects:	GCC_VERSION_IDENTIFIER.

GCC_VERSION_IDENTIFIER

Description:	Identifier. Identifies the version of GCC to be used to compile the target's source files. This build setting is unavailable in Run Script build phases when GCC_VERSION is not available in them.
Effectors:	GCC_VERSION
Value:	The value of GCC_VERSION using underscores instead of periods.
Example value:	4_0

GCC_WARN_ABOUT_GLOBAL_CONSTRUCTORS (Global Construction or Destruction Required)

Description:	Boolean value. Specifies whether to warn about the use of static initializers.
Values:	• YES: Warn about the use of static initializers (gcc -Wglobal-constructors). • NO: Do not warn about the use of static initializers.

Default value:	`NO`

GCC_WARN_ABOUT_RETURN_TYPE (Mismatched Return Type)

Description:	Boolean value. Specifies whether to warn about functions that do not have an explicit return type and about functions that contain `return` statements but whose return type is `void`.
Values:	▪ `YES`: Warn about ambiguous function return types (`gcc -Wreturn-type`). ▪ `NO`: Do not warn about ambiguous function return types.
Default value:	`NO`

GCC_WARN_UNUSED_VARIABLE (Unused Variables)

Description:	Boolean value. Specifies whether warn about unused local variables or unused nonconstant static variables.
Values:	▪ `YES`: Warn about unused variables (`gcc - Wunused-variable`). ▪ `NO`: Do not warn about unused variables.
Default value:	`NO`

GCC_WARN_EFFECTIVE_CPLUSPLUS_VIOLATIONS (Effective C++ Violation)

Description:	Boolean value. Specifies whether to warn about violations to certain code style guidelines described in *Effective C++* (by Scott Meyer).
Values:	▪ `YES`: Warn about *Effective C++*-style violations (`gcc -Weffc++`). ▪ `NO`: Do not warn about *Effective C++*-style violations.
Default value:	`NO`

GCC_WARN_HIDDEN_VIRTUAL_FUNCTIONS (Hidden Virtual Functions)

Description:	Boolean value. Specifies whether to warn about function declarations that hide virtual functions declared in a base class.
Values:	▪ `YES`: Warn about function declarations that hide virtual functions declared in a base class (`gcc -Woverloaded-virtual`). ▪ `NO`: Do not warn about function declarations that hide virtual functions declared in a base class
Default value:	`NO`

GCC_WARN_INHIBIT_ALL_WARNINGS (Inhibit All Warnings)

Description:	Boolean value. Specifies whether to suppress warnings.

Values:	▪ YES: Suppress all warnings (gcc -w). ▪ NO: Do not suppress warnings.
Default value:	NO

GCC_WARN_NON_VIRTUAL_DESTRUCTOR (Nonvirtual Destructor)

Description:	Boolean value. Specifies whether to warn about classes that declare a nonvirtual destructor that should be virtual (when the compiler determines that the class is used polymorphically). This build setting applies only to C++ and Objective-C++ source files.
Values:	▪ YES: Warn about nonvirtual destructors that should be virtual (gcc -Wnon-virtual-dtor). ▪ NO: Do not warn about nonvirtual destructors that should be virtual.
Default value:	NO

GCC_WARN_PEDANTIC (Pedantic Warnings)

Description:	Boolean value. Specifies whether to warn about source code that does not adhere to ISO C or ISO C++ standards.
Values:	▪ YES: Warn about nonadherence to ISO C or ISO C++ standards (gcc -pedantic). ▪ NO: Do not warn about nonadherence to ISO C or ISO C++ standards.
Default value:	NO

GCC_WARN_SHADOW (Hidden Local Variables)

Description:	Boolean value. Specifies whether to warn about local symbols that shadow another local variable, parameter, or global variable, built-in function.
Values:	▪ YES: Warn about shadowed symbols (gcc -Wshadow). ▪ NO: Do not warn about shadowed symbols.
Default value:	NO

GCC_WARN_SIGN_COMPARE (Sign Comparison)

Description:	Boolean value. Specifies whether to warn about comparisons between signed and unsigned values that could produce an incorrect result when the signed value is converted to unsigned.
Values:	▪ YES: Warn about sign discrepancies in comparisons (gcc -Wsign-compare). ▪ NO: Do not warn about sign discrepancies in comparisons.
Default value:	NO

HEADER_SEARCH_PATHS (Header Search Paths)

Description:	Space-separated list of directory paths. Specifies directories in which to search for header files. (In GCC, this list is passed in the `gcc -I` option.) When this build setting is defined, `$SDKROOT` is added to the beginning of each system-header path passed to the compiler.
Default value:	None.
Example values:	`/Users/genica/TestHeaders/**` `/System/Library/Frameworks/AddressBook.framework`
Companion:	SDKROOT (Base SDK).
Related to:	USER_HEADER_SEARCH_PATHS (User Header Search Paths).

INFOPLIST_OTHER_PREPROCESSOR_FLAGS (Info.plist Other Preprocessor Flags)

Description:	Space-separated list of option specifications. Specifies additional options for preprocessing the info plist file.
Companion:	INFOPLIST_PREPROCESS (Preprocess Info.plist File), INFOPLIST_FILE.
Related to:	INFOPLIST_PREFIX_HEADER (Info.plist Preprocessor Prefix File), INFOPLIST_PREPROCESSOR_DEFINITIONS (Info.plist Preprocessor Definitions).

INFOPLIST_PREFIX_HEADER (Info.plist Preprocessor Prefix File)

Description:	File path or project file path. Specifies the path to the prefix file to include when processing the info plist file.
Companion:	INFOPLIST_PREPROCESS (Preprocess Info.plist File).

INFOPLIST_PREPROCESS (Preprocess Info.plist File)

Description:	Boolean. Specifies whether to preprocess the info plist file.
Values:	▪ `YES`: Preprocesses the info plist file. ▪ `NO`: Doesn't preprocess the info plist file.
Default value:	`NO`.
Companion:	INFOPLIST_FILE.
Related to:	INFOPLIST_PREFIX_HEADER (Info.plist Preprocessor Prefix File).

INFOPLIST_PREPROCESSOR_DEFINITIONS (Info.plist Preprocessor Definitions)

Description:	Space-separated list of option specifications. Defines preprocessor macros used when preprocessing the info plist file.
Example value:	`DEBUG=1`.
Companion:	INFOPLIST_PREPROCESS (Preprocess Info.plist File), INFOPLIST_FILE.
Related to:	

	INFOPLIST_OTHER_PREPROCESSOR_FLAGS (Info.plist Other Preprocessor Flags).

IPHONEOS_DEPLOYMENT_TARGET (iOS Deployment Target)

Description:	Numeric identifier. Identifies the earliest iOS version the product is to run on. This build setting is available in Run Script build phases only when it is set to a specific iOS version.
Values:	`2.0`: Product runs on iOS 2.0 and later. `2.1`: Product runs on iOS 2.1 and later. `2.2`: Product runs on iOS 2.2 and later. `2.2.1`: Product runs on iOS 2.2.1 and later. `3.0`: Product runs on iOS 3.0 and later. `3.1`: Product runs on iOS 3.1 and later. `3.1.2`: Product runs on iOS 3.1.2 and later. `3.1.3`: Product runs on iOS 3.1.3 and later. `3.2`: Product runs on iOS 3.2 and later. `4.0`: Product runs on iOS 4.0 and later.
Default value:	Compiler default. Product runs on the iOS version `SDKROOT` targets, and later.
Related to:	SDKROOT (Base SDK).

MACOSX_DEPLOYMENT_TARGET (OS X Deployment Target)

Description:	Numeric identifier. Identifies the earliest OS X version the product is to run on. This build setting is available in Run Script build phases only when it is set to a specific OS X version.
Values:	`10.1`: Product runs on 10.1 if no 10.2 or 10.3 API is used, on 10.2 with weak linking, and on 10.3 or later fully linked. `10.2`: Product runs on 10.2 with weak linking, and on 10.3 or later fully linked. `10.3`: Product runs only on 10.3 and later. `10.4`: Product runs only on 10.4 and later. `10.5`: Product runs only on 10.5 and later. `10.6`: Product runs only on 10.6 and later.
Default value:	Compiler default. Product runs on the OS X version `SDKROOT` targets, and later.
Related to:	SDKROOT (Base SDK).

OTHER_CFLAGS (Other C Flags)

Description:	Space-separated list of option specifications. Specifies additional options for compiling C-based precompiled headers and implementation files. These options are passed (as given) to the compiler whether other build settings also specify values that correspond to these options. Therefore, you should look for the appropriate compiler build setting to specify a particular compiler option before using this build setting.
Default value:	None.
Example value:	`-dM`
Affects:	OTHER_CPLUSPLUSFLAGS (Other C++ Flags).

Related to:	OTHER_CFLAGS_<VARIANT>.

OTHER_CFLAGS_<VARIANT>

Description:	Space-separated list of option specifications. Specifies additional options for compiling C-based (including C++) precompiled headers and implementation files for the specified variant. These options are passed (as given) to the compiler whether other build settings also specify values that correspond to these options. Therefore, you should look for the appropriate compiler build setting to specify a particular compiler option before using this build setting.
Default value:	None.
Related to:	BUILD_VARIANTS (Build Variants), OBJECT_FILE_DIR_<VARIANT>, OTHER_CFLAGS (Other C Flags), OTHER_CPLUSPLUSFLAGS (Other C++ Flags).

OTHER_CPLUSPLUSFLAGS (Other C++ Flags)

Description:	Space-separated list of option specifications. Specifies additional options for compiling C++-based precompiled headers and implementation files. These options are passed (as given) to the compiler whether other build settings also specify values that correspond to these options. Therefore, you should look for the appropriate compiler build setting to specify a particular compiler option before using this build setting.
Effectors:	OTHER_CFLAGS (Other C Flags).
Default value:	`$OTHER_CFLAGS`.
Example value:	`-Weffc++`
Related to:	OTHER_CFLAGS_<VARIANT>.

USER_HEADER_SEARCH_PATHS (User Header Search Paths)

Description:	Space-separated list of directory paths. Specifies directories to search for header files included in source files using quotation marks ("") instead of angle brackets (<>). User header files are supported in GCC 4.0 and later. Relative paths are relative to the project directory (SRCROOT). Xcode build tools, such as GCC, are invoked with their working directory set to `SRCROOT`. Third-party build tools should take care not to change the working directory; otherwise, the relative search paths passed to them may produce unexpected results.
Default value:	None.
Companion:	SRCROOT.
Related to:	HEADER_SEARCH_PATHS (Header Search Paths).

WARNING_CFLAGS (Other Warning Flags)

Description:	Space-separated list of option specifications. Specifies additional warning options for compiling C-based (including C++) precompiled headers and implementation files. These options are passed (as given) to the compiler whether other build settings also specify values that correspond to these options. Therefore, you should look for the appropriate compiler build setting to specify a particular warning option before using this build setting.
Default	None.

value:	
Related to:	`GCC_WARN…` build settings.

Interface Builder Compiler Build Settings

These build settings specify options for the Interface Builder compiler.

IBC_FLATTEN_NIBS (Flatten Compiles XIB Files)

Description:	Boolean value. Specifies whether to strip a nib files to reduce their size. The resulting nib file is more compact but is not editable.
Values:	▪ `YES`: Strips nib files, making them uneditable. ▪ `NO`: Does not strip nib files.
Default value:	`YES`.

IBC_OTHER_FLAGS (Other Interface Builder Compiler Flags)

Description:	Space-separated list of option specifications. Specifies additional options for compiling nib files. These options are passed (as given) to the compiler whether other build settings also specify values that correspond to these options. Therefore, you should look for the appropriate compiler build setting to specify a particular compiler option before using this build setting.
Default value:	None.

IBC_OVERRIDING_PLUGINS_AND_FRAMEWORKS_DIR (Overriding Plug-In and Framework Directory)

Description:	Directory path. Identifies the directory from which to load frameworks and Interface Builder plug-ins.
Example value:	`$(BUILD_DIR)/$(CONFIGURATION)$(EFFECTIVE_PLATFORM_NAME)`. Loads frameworks and plug-ins from the built products directory of the active build configuration.
Default value:	None.

IBC_PLUGIN_SEARCH_PATHS (Plug-In Search Paths)

Description:	Space-separated list of directory paths. Identifies directories to be searched for Interface Builder plug-ins to load when compiling xib files.
Default value:	None.

IBC_PLUGINS (Plug-Ins)

Description:	Space-separated list of file paths. Identifies Interface Builder plug-ins to load when compiling xib files.
Default value:	None.

IBC_ERRORS (Show Errors)

Description:	Boolean value. Specifies whether to emit xib-file-compilation errors.
Values:	▪ `YES`: Emits xib-file-compilation errors. ▪ `NO`: Does not emit xib-file-compilation errors.
Default value:	`YES`.

IBC_NOTICES (Show Notices)

Description:	Boolean value. Specifies whether to emit xib-file-compilation notices.
Values:	▪ `YES`: Emits xib-file-compilation notices. ▪ `NO`: Does not emit xib-file-compilation notices.
Default value:	`YES`.

IBC_WARNINGS (Show Warnings)

Description:	Boolean value. Specifies whether to emit xib-file-compilation warnings.
Values:	▪ `YES`: Emits xib-file-compilation warnings. ▪ `NO`: Does not emit xib-file-compilation warnings.
Default value:	`YES`.

Linker Build Settings

These build settings specify linking options.

DEAD_CODE_STRIPPING (Dead Code Stripping)

Description:	Boolean value. Specifies whether dead code is stripped from the binary.
Prerequisite:	`$GCC_DEBUGGING_SYMBOLS = full`
Values:	▪ `YES`: Dead code is stripped from the binary when the prerequisite is met. ▪ `NO`: Dead code is not stripped from the binary.
Default value:	`NO`
Companions:	GCC_DEBUGGING_SYMBOLS (Level of Debug Symbols).

Prerequisite for:	PRESERVE_DEAD_CODE_INITS_AND_TERMS

EXPORTED_SYMBOLS_FILE (Exported Symbols File)

Description:	Project file path. Identifies a file containing the names of global symbols to be exported from the binary. All other symbols are treated as if they had been marked as private. See Minimizing Your Exported Symbols in *Code Size Performance Guidelines* and `ld -exported_symbols_list` for details on exporting symbols.
Default value:	None.
Example value:	`My_Public_Symbols`

KEEP_PRIVATE_EXTERNS (Preserve Private External Symbols)

Description:	Boolean value. Specifies whether private external symbols remain so in the binary.
Values:	▪ `YES`: Private external symbols in source code are private external in the binary (`ld -keep_private_externs`). ▪ `NO`: Private external symbols in source code are static symbols in the binary.
Default value:	`NO`

LD_DYLIB_INSTALL_NAME (Dynamic Library Install Name)

Description:	File path. Specifies the install name of a dynamic library. See *Dynamic Library Programming Topics*.
Default value:	None.
Example values:	▪ `/usr/lib/libfoo` ▪ `@rpath/libfoo`
Related to:	LD_RUNPATH_SEARCH_PATHS (Runpath Search Paths).

LD_RUNPATH_SEARCH_PATHS (Runpath Search Paths)

Description:	Space-separated list of directory paths. Specifies the run-path locations at which the dynamic loader searches for the product's run-path dependent libraries. See *Dynamic Library Programming Topics*.
Default value:	None.
Example values:	`@loader_path/../Frameworks` `/usr/lib`
Related to:	LD_DYLIB_INSTALL_NAME (Dynamic Library Install Name).

LIBRARY_SEARCH_PATHS

Description:	Space-separated list of directory paths. Specifies directories in which the linker searches for included libraries to link the binary against. Adding `**` to the end of a path specifies a recursive path. When this build setting is defined, `$SDKROOT` is added to the beginning of each path passed to the linker.
Default value:	None.
Example value:	`/Volumes/Sauron/Team/Libs`
Companion:	SDKROOT (Base SDK).

LINK_WITH_STANDARD_LIBRARIES (Link With Standard Libraries)

Description:	Boolean value. Specifies whether to link the binary against the standard libraries. When not linking against the standard libraries, you should use OTHER_LDFLAGS (Other Linker Flags) to specify the libraries to link binary against.
Values:	▪ `YES`: Binary is linked against standard libraries. ▪ `NO`: Binary is not linked against standard libraries.
Default value:	`YES`

LINKER_DISPLAYS_FILE_FOR_UNDEFINED_SYMBOLS (Verbose Undefined Symbols Info)

Description:	Boolean value. Specifies whether the linker displays additional information about undefined symbols, such as the source file the symbol is used in and whether the file references or defines the symbol.
Values:	▪ `YES`: The linker displays additional information about undefined symbols (`ld -Y`). ▪ `NO`: The linker does not display additional information about undefined symbols.
Default value:	`YES`

LINKER_DISPLAYS_MANGLED_NAMES (Display Mangled Names)

Description:	Boolean value. Specifies whether the linker displays mangled names for C++ symbols. This information can help in diagnosing C++ linking problems.
Values:	▪ `YES`: The linker displays mangled names for C++ symbols (`ld --no-demangle`). ▪ `NO`: The linker does not display mangled names for C++ symbols.
Default value:	`NO`

OTHER_LDFLAGS (Other Linker Flags)

Description:	Space-separated list of option specifications. Specifies additional options for linking the binary. These options are passed (as given) to the linker whether other build settings also specify values that correspond to these options. Therefore, you should look for the appropriate linker build setting to specify a particular linker option before using this build setting.
Default	None.

value:	
Related to:	OTHER_LDFLAGS_<VARIANT>.

OTHER_LDFLAGS_<VARIANT>

Description:	Space-separated list of option specifications. Specifies additional options for linking the binary for the specified variant. These options are passed (as given) to the linker whether other build settings also specify values that correspond to these options. Therefore, you should look for the appropriate linker build setting to specify a particular linker option before using this build setting.
Default value:	None.
Related to:	OTHER_LDFLAGS (Other Linker Flags).

PREBINDING (Prebinding)

Description:	Boolean value. Specifies whether to prebind the generated binary.
Prerequisite:	`($ARCHS * $VALID_ARCHS) IN {ppc, ppc970}`
Values:	▪ `YES`: The binary is prebound when the prerequisite is met. ▪ `NO`: The binary is not prebound.
Default value:	`YES`
Companion:	ARCHS (Architectures), VALID_ARCHS (Valid Architectures).

PRESERVE_DEAD_CODE_INITS_AND_TERMS (Don’t Dead-Strip Inits and Terms)

Description:	Boolean value. Specifies whether to prevent initialization and termination routines from being dead-code stripped.
Prerequisite:	`$DEAD_CODE_STRIPPING = YES`
Values:	▪ `YES`: Prevents dead-code stripping of initializers and terminators when the prerequisite is met. ▪ `NO`: Does not prevent dead-code stripping of initializers and terminators.
Default value:	`NO`
Companion:	DEAD_CODE_STRIPPING (Dead Code Stripping).

STANDARD_C_PLUS_PLUS_LIBRARY_TYPE (C++ Standard Library Type)

Description:	Identifier. Specifies how the binary is linked against the C++ standard library: As a dynamic library or as a static library.
Prerequisite:	`$GCC_SYMBOLS_PRIVATE_EXTERN = YES`. See for details.
Values:	`dynamic`: The C++ standard library is linked as a dynamic library. `static`: The C++ standard library is linked as a static library when the prerequisite is met.
Default value:	`dynamic`

Companion:	GCC_SYMBOLS_PRIVATE_EXTERN (Symbols Hidden by Default).

STRIP_INSTALLED_PRODUCT (Strip Linked Product)

Description:	Boolean value. Specifies whether to strip symbol information from the binary.
Prerequisite:	`$DEPLOYMENT_POSTPROCESSING = YES`
Values:	▪ `YES`: Strips the generated binary when the prerequisite is met. ▪ `NO`: Does not strip the generated binary.
Default value:	`NO`
Companion:	DEPLOYMENT_POSTPROCESSING (Deployment Postprocessing).
Related to:	STRIP_STYLE (Strip Style).

STRIP_STYLE (Strip Style)

Description:	Identifier. Specifies the level of stripping performed on the binary.
Values:	▪ `all`: Strips the binary completely, removing the symbol table and relocation information. ▪ `non-global`: Strips nonglobal symbols but saves external symbols. ▪ `debugging`: Strips debugging symbols but saves local and global symbols.
Default value:	▪ `all`: Application and command-line products. ▪ `non-global`: Bundle products. ▪ `debugging`: Library and framework products.
Related to:	STRIP_INSTALLED_PRODUCT (Strip Linked Product).

UNEXPORTED_SYMBOLS_FILE (Unexported Symbols File)

Description:	Project file path. Identifies a file containing the names of global symbols to be hidden. See Minimizing Your Exported Symbols in *Code Size Performance Guidelines* and `ld -exported_symbols_list` for details on exporting symbols.
Default value:	None.
Example value:	`My_Private_Symbols`

Product Layout Build Settings

These build settings specify the layout of bundle-based products.

CONTENTS_FOLDER_PATH

Description:	Bundle directory path. Specifies the directory inside the generated bundle that contains the product's files.

Effector:	WRAPPER_NAME.
Default value:	`$WRAPPER_NAME/Contents`
Example value:	`MyProduct.bundle/Contents`
Affects:	EXECUTABLE_PATH, FRAMEWORKS_FOLDER_PATH, INFOPLIST_PATH, PLUGINS_FOLDER_PATH, PRIVATE_HEADERS_FOLDER_PATH, PUBLIC_HEADERS_FOLDER_PATH, SCRIPTS_FOLDER_PATH, SHARED_FRAMEWORKS_FOLDER_PATH, UNLOCALIZED_RESOURCES_FOLDER_PATH.

INFOPLIST_FILE

Description:	Filename. Specifies the name of the information property list file that specifies the bundled product's runtime properties. For details on information property list files, see Information Property List Files in*Runtime Configuration Guidelines*. You should not change the value of this build setting from its default. Doing so produces a bundled product that may not work as expected in OS X.
Default value:	`Info.plist`
Affects:	INFOPLIST_PATH.
Related to:	INFOPLIST_OUTPUT_FORMAT

INFOPLIST_OUTPUT_FORMAT

Description:	Identifier. Specifies the whether the information property list file is written using the binary format.
Values:	▪ `binary`: Specifies the binary format. ▪ `<unspecified>`: Specifies the XML-based format.
Related to:	INFOPLIST_FILE.

INFOPLIST_PATH

Description:	Bundle file path. Specifies the path to the bundle's information property list file.
Effectors:	INFOPLIST_FILE, CONTENTS_FOLDER_PATH.
Default value:	`$CONTENTS_FOLDER_PATH/$INFOPLIST_FILE`
Example value:	`MyProduct.bundle/Contents/Info.plist`

INFOSTRINGS_PATH

Description:	Bundle file path. Specifies the file that contains the bundle's localized strings file.
Default value:	`/InfoPlist.strings`

FRAMEWORKS_FOLDER_PATH

Description:	Bundle directory path. Specifies the directory that contains the product's embedded frameworks.
Effector:	CONTENTS_FOLDER_PATH.
Default value:	`$CONTENTS_FOLDER_PATH/Contents/Frameworks`
Example value:	`MyProduct.bundle/Contents/Frameworks`

GENERATE_PKGINFO_FILE

Description:	Boolean value. Specifies whether to generate the file specified by `PKGINFO_FILE_PATH`, even when the file is not expected.
Values:	▪ `YES`: Always generate the package information file. ▪ `NO`: Do not generate the package information file.
Default value:	▪ `YES`: In application targets. ▪ `NO`: In other target types.
Companion:	PKGINFO_FILE_PATH.

DOCUMENTATION_FOLDER_PATH

Description:	Bundle directory path. Identifies the directory that contains the bundle's documentation files.
Default value:	`/Documentation`

EXECUTABLES_FOLDER_PATH

Description:	Bundle directory path. Identifies the directory that contains additional binary files.
Effector:	CONTENTS_FOLDER_PATH.
Default value:	`$CONTENTS_FOLDER_PATH/Executables`
Example value:	`MyProduct.bundle/Contents/Executables`

EXECUTABLE_EXTENSION

Description:	Identifier. Specifies the extension of the binary the target produces.
Effectors:	MACH_O_TYPE
Default values:	▪ `bundle`: When `$MACH_O_TYPE = mh_bundle`. ▪ `dylib`: When `$MACH_O_TYPE = mh_dylib`. ▪ `a`: When `$MACH_O_TYPE = staticlib`. ▪ *none*: When `$MACH_O_TYPE = mh_executable`.
Affects:	EXECUTABLE_SUFFIX.

EXECUTABLE_FOLDER_PATH

Description:	Bundle directory path. Identifies the directory that contains the binary the target builds.
Effector:	CONTENTS_FOLDER_PATH.
Default value:	`$CONTENTS_FOLDER_PATH/MacOS`
Example value:	`MyProduct.app/Contents/MacOS`

EXECUTABLE_NAME

Description:	Filename. Specifies the name of the binary the target produces.
Effectors:	PRODUCT_NAME, EXECUTABLE_PREFIX, EXECUTABLE_SUFFIX.
Default value:	`$EXECUTABLE_PREFIX$PRODUCT_NAME$EXECUTABLE_SUFFIX`
Example values:	▪ `MyProduct` ▪ `MyDynamicLibrary.dylib`
Affects:	EXECUTABLE_PATH.

EXECUTABLE_PATH

Description:	Bundle directory path. Specifies the path to the binary the target produces within its bundle.
Effectors:	CONTENTS_FOLDER_PATH, , EXECUTABLE_NAME.
Default value:	`$CONTENTS_FOLDER_PATH$EXECUTABLE_NAME`
Example values:	▪ `MyApp.app/Contents/MacOS/MyApp` ▪ `MyDynamicLibrary.dylib`

EXECUTABLE_PREFIX

Description:	File prefix. Specifies the prefix of the binary filename.
Default value:	None.
Affects:	EXECUTABLE_NAME.

EXECUTABLE_SUFFIX

Description:	File suffix. Specifies the suffix of the binary filename (including the character that separates the extension from the rest of the bundle name).
Effector:	EXECUTABLE_EXTENSION.

Default value:	`.$EXECUTABLE_EXTENSION`
Example value:	`.bundle`
Affects:	EXECUTABLE_NAME.

PACKAGE_TYPE

Description:	Uniform type identifier. Identifies the type of the product the target builds. Some products may be made up of a single binary or archive. Others may comprise several files, which are grouped under a single directory. These container directories are known as *bundles*.
Value:	`com.apple.package-type.wrapper`: In kernel extension, application, bundle, and plug-in targets. `com.apple.package-type.wrapper.framework`: In framework targets. `com.apple.package-type.mach-o-executable`: In command-line utility targets. `com.apple.package-type.mach-o-dylib`: In dynamic library targets. `com.apple.package-type.static-library`: In static library targets.

PLUGINS_FOLDER_PATH

Description:	Bundle directory path. Specifies the directory that contains the product's plug-ins.
Effector:	CONTENTS_FOLDER_PATH.
Default value:	`$CONTENTS_FOLDER_PATH/Contents/PlugIns`
Example value:	`MyProduct.bundle/Contents/PlugIns`

PRIVATE_HEADERS_FOLDER_PATH

Description:	Bundle directory path. Specifies the directory that contains the product's private header files.
Effector:	CONTENTS_FOLDER_PATH.
Default value:	`$CONTENTS_FOLDER_PATH/Contents/PrivateHeaders`
Example value:	`MyProduct.bundle/Contents/PrivateHeaders`

PKGINFO_FILE_PATH

Description:	Bundle file path. Specifies the file that contains the bundle's package information file.
Effector:	CONTENTS_FOLDER_PATH.
Value:	`$CONTENTS_FOLDER_PATH/PkgInfo`
Example value:	`MyProduct.bundle/Contents/PkgInfo`

PUBLIC_HEADERS_FOLDER_PATH

Description:	Bundle directory path. Specifies the directory that contains the product's public header files.
Effector:	CONTENTS_FOLDER_PATH.
Default value:	`$CONTENTS_FOLDER_PATH/Contents/PublicHeaders`
Example value:	`MyProduct.bundle/Contents/PublicHeaders`

SCRIPTS_FOLDER_PATH

Description:	Bundle directory path. Specifies the directory that contains the product's scripts.
Effector:	CONTENTS_FOLDER_PATH.
Default value:	`$(UNLOCALIZED_RESOURCES_FOLDER_PATH)/Scripts`
Example value:	`MyProduct.bundle/Contents/Resources/Scripts`

SHARED_FRAMEWORKS_FOLDER_PATH

Description:	Bundle directory path. Specifies the directory that contains the product's shared frameworks.
Effector:	CONTENTS_FOLDER_PATH.
Default value:	`$CONTENTS_FOLDER_PATH/Contents/SharedFrameworks`
Example value:	`MyProduct.bundle/Contents/SharedFrameworks`

UNLOCALIZED_RESOURCES_FOLDER_PATH

Description:	Bundle directory path. Specifies the directory that contains the product's unlocalized resources.
Effector:	CONTENTS_FOLDER_PATH.
Default value:	`$CONTENTS_FOLDER_PATH/Contents/Resources`
Example value:	`MyProduct.bundle/Contents/Resources`

WRAPPER_EXTENSION (Wrapper Extension)

Description:	Identifier. Specifies the extension of the product bundle name (not including the character that separates the extension from the rest of the bundle name).
Effector:	Product type choose when the target was created
Default value:	`app`: In application products. `kext`: In kernel extension products. `bundle`: In bundle and plug-in products. `framework`: In framework products. *none*: In command-line utility, dynamic library, and static library products.

Example value:	`bundle`
Affects:	WRAPPER_SUFFIX.

WRAPPER_NAME

Description:	Filename. Specifies the filename (including the appropriate extension) of the product bundle.
Effectors:	PRODUCT_NAME, WRAPPER_SUFFIX.
Value:	`$PRODUCT_NAME.$WRAPPER_SUFFIX`
Example value:	`MyProduct.bundle`
Related to:	PACKAGE_TYPE.

WRAPPER_SUFFIX

Description:	File suffix. Specifies the suffix of the product bundle name (including the character that separates the extension from the rest of the bundle name).
Effector:	WRAPPER_EXTENSION (Wrapper Extension).
Default value:	`.$WRAPPER_EXTENSION`
Example value:	`.bundle`
Affects:	WRAPPER_NAME.

Code Signing Build Settings

These build settings specify code signing options.

CODE_SIGN_ENTITLEMENTS (Code Signing Entitlements)

Description:	Filename. Specifies the name of the application's entitlements property-list file. This build setting applies only to iOS applications.
Example value:	`Entitlements.plist`

CODE_SIGN_IDENTITY (Code Signing Identity)

Description:	Identifier. Specifies the name of a code signing identity.
Example value:	`iPhone Developer`

CODE_SIGN_RESOURCE_RULES_PATH (Code Signing Resource Rules Path)

Description:	File path. Identifies a property-list file containing resource-scanning instructions that override the rules for identifying bundle resources to sign.
Example value:	`ResourceRules.plist`

OTHER_CODE_SIGN_FLAGS (Other Code Signing Flags)

Description:	Space-separated list of option specifications. Specifies additional options to `codesign` for code-signing binaries.
Example value:	`-i MySigningIdentifier`

Copy Build Settings

These build settings specify file-copying options.

COPY_PHASE_STRIP (Strip Debug Symbols During Copy)

Description:	Boolean value. Specifies whether copied binaries are stripped of debugging symbols.
Values:	▪ `YES`: Copied binaries are stripped of debugging symbols. This does not cause the binary produced by the linker to be stripped. Use STRIP_INSTALLED_PRODUCT (Strip Linked Product) to have the linker strip the binary. ▪ `NO`: Copied binaries are not stripped of debugging symbols
Default value:	`NO`

INSTALLHDRS_COPY_PHASE

Description:	Boolean value. Specifies whether the target's Copy Files build phases are executed in install-header builds.
Values:	▪ `YES`: Copy Files build phases are executed in install-header builds. ▪ `NO`: Copy Files build phases are not executed in install-header builds.
Default value:	`NO`
Companion:	ACTION.

INSTALLHDRS_SCRIPT_PHASE

Description:	Boolean value. Specifies whether the target's Run Script build phases are executed in install-header builds. See ACTION for details on install-header builds.
Values:	▪ `YES`: Run Script build phases are executed in install-header builds. ▪ `NO`: Run Script build phases are not executed in install-header builds.
Default	

value:	NO
Companion:	ACTION.

REMOVE_CVS_FROM_RESOURCES

Description:	Boolean value. Specifies whether to remove CVS directories from bundle resources when they are copied.
Values:	▪ YES: CVS directories are removed from copied bundle resources. ▪ NO: CVS directories are not removed from copied bundle resources.
Default value:	YES

REMOVE_SVN_FROM_RESOURCES

Description:	Boolean value. Specifies whether to remove SVN directories from bundle resources when they are copied.
Values:	▪ YES: SVN directories are removed from copied bundle resources. ▪ NO: SVN directories are not removed from copied bundle resources.
Default value:	YES

VERBOSE_PBXCP

Description:	Boolean value. Specifies whether the target's Copy Files build phases generate additional information when copying files.
Values:	▪ YES: Copy Files build phases generate additional information. ▪ NO: Copy Files build phases do not generate additional information.
Default value:	NO

User Location Build Settings

These build settings represent locations in the User realm in the filesystem.

HOME

Description:	File path. Specifies the path to the user's home directory.
Value:	~: Fully qualified path to a user's home directory.
Example:	/Users/genica
Affects:	INSTALL_PATH (Installation Directory).

USER_LIBRARY_DIR

Description:	File path. Specifies the path the user's `Library` directory.
Value:	`~/Library`: Fully qualified path to the user's `Library` directory.
Affects:	INSTALL_PATH (Installation Directory).

System Location Build Setting

This build setting represents a location in the System realm in the file system.

SYSTEM_LIBRARY_DIR

Description:	Directory path. Specifies the path of the `/System/Library` directory.
Value:	`/System/Library`
Affects:	INSTALL_PATH (Installation Directory).