NAME

xcodebuild — build Xcode projects and workspaces

SYNOPSIS

xcodebuild [**-project** *name.xcodeproj*] [[**-target** *targetname*] ... | **-alltargets**]
  [**-configuration** *configurationname*]
  [**-sdk** [*sdkfullpath* | *sdkname*]] [*action* ...]
  [*buildsetting=value* ...] [*-userdefault=value* ...]

xcodebuild [**-project** *name.xcodeproj*] **-scheme** *schemename*
  [[**-destination** *destinationspecifier*] ...]
  [**-destination-timeout** *value*]
  [**-configuration** *configurationname*]
  [**-sdk** [*sdkfullpath* | *sdkname*]] [*action* ...]
  [*buildsetting=value* ...] [*-userdefault=value* ...]

xcodebuild **-workspace** *name.xcworkspace* **-scheme** *schemename*
  [[**-destination** *destinationspecifier*] ...]
  [**-destination-timeout** *value*]
  [**-configuration** *configurationname*]
  [**-sdk** [*sdkfullpath* | *sdkname*]] [*action* ...]
  [*buildsetting=value* ...] [*-userdefault=value* ...]

xcodebuild **-version** [**-sdk** [*sdkfullpath* | *sdkname*]] [*infoitem*]

xcodebuild **-showsdks**

xcodebuild **-showBuildSettings**
  [**-project** *name.xcodeproj* | [**-workspace** *name.xcworkspace* **-scheme** *schemename*]

xcodebuild **-list** [**-project** *name.xcodeproj* | **-workspace** *name.xcworkspace*]

xcodebuild **-exportArchive** **-archivePath** *xcarchivepath* **-exportPath**
  destinationpath **-exportOptionsPlist** *path*

xcodebuild **-exportLocalizations** **-project** *name.xcodeproj* **-localizationPath**
  path [[**-exportLanguage** *language*] ...]
xcodebuild **-importLocalizations** **-project** *name.xcodeproj* **-localizationPath**
  path

DESCRIPTION

xcodebuild builds one or more targets contained in an Xcode project, or builds a scheme contained in an Xcode workspace or Xcode project.

Usage

To build an Xcode project, run **xcodebuild** from the directory containing your project (i.e. the directory containing the *name.xcodeproj* package) . If you have multiple projects in the this directory you will need to use **-project** to indicate which project should be built. By default, **xcodebuild** builds the first target listed in the project, with the default build configuration. The order of the targets is a property of the project and is the same for all users of the project.

To build an Xcode workspace, you must pass both the **-workspace** and **-scheme** options to define the build. The parameters of the scheme will control which targets are built and how they are built, although you may pass other options to **xcodebuild** to override some parameters of the scheme.

There are also several options that display info about the installed version of Xcode or about projects or workspaces in the local directory, but which do not initiate an action. These include **-list**, **-showBuildSettings,** **-showsdks**, **-usage**, and **-version**.

Options

-project *name.xcodeproj*
Build the project *name.xcodeproj*. Required if there are multiple project files in the same directory.

-target *targetname*
Build the target specified by *targetname*.

-alltargets
Build all the targets in the specified project.

-workspace *name.xcworkspace*
Build the workspace *name.xcworkspace*.

-scheme *schemename*
Build the scheme specified by *schemename*. Required if building a workspace.

-destination *destinationspecifier*
Use the destination device described by *destinationspecifier*. Defaults to a destination that is compatible with the selected scheme. See the **Destinations** section below for more details.

-destination-timeout *timeout*
Use the specified *timeout* when searching for a destination device. The default is 30 seconds.

-configuration *configurationname*
Use the build configuration specified by *configurationname* when building each target.

-arch *architecture*
Use the architecture specified by *architecture* when building each target.

-sdk [*sdkfullpath* | *sdkname*]
Build an Xcode project or workspace against the specified SDK, using build tools appropriate for that SDK. The argument may be an absolute path to an SDK, or the canonical name of an SDK.

-showsdks
Lists all available SDKs that Xcode knows about, including their canonical names suitable for use with **-sdk**. Does not initiate a build.

-showBuildSettings
Lists the build settings in a project or workspace and scheme. Does not initiate a build. Use with **-project** or **-workspace** and **-scheme.**

-list
Lists the targets and configurations in a project, or the schemes in a workspace. Does not initiate a build. Use with **-project** or **-workspace**.

-enableAddressSanitizer [*YES* | *NO*]
Turns the address sanitizer on or off. This overrides the setting for the launch action of a scheme in a workspace.

-enableCodeCoverage [*YES* | *NO*]
Turns code coverage on or off during testing. This overrides the setting for the test action of a scheme in a workspace.

-derivedDataPath *path*
Overrides the folder that should be used for derived data when performing an action on a scheme in a workspace.

-resultBundlePath *path*
Writes a bundle to the specified *path* with results from performing an action on a scheme in a workspace.

-exportArchive
Specifies that an archive should be exported. Requires **-archivePath**, **-exportPath**, and **-exportOptionsPlist**. Cannot be passed along with an action.

-archivePath *xcarchivepath*
Specifies the path for the archive produced by the `archive` action, or specifies the archive that should be exported when **-exportArchive** is passed.

-exportPath *destinationpath*
Specifies the destination for the exported product, including the name of the exported file.

-exportOptionsPlist *path*
Specifies options for **-exportArchive**. **xcodebuild -help** can print the full set of available options.

-exportLocalizations
Exports localizations to XLIFF files. Requires -project and -localizationPath. Cannot be passed along with an action.

-importLocalizations
Imports localizations from an XLIFF file. Requires -project and -localizationPath. Cannot be passed along with an action.

-localizationPath
Specifies a path to a directory or a single XLIFF localization file.

-exportLanguage *language*
Specifies optional ISO 639-1 languages included in a localization export. May be repeated to specify multiple languages. May be excluded to specify an export includes only development language strings.

action ...
Specify one or more actions to perform. Available actions are:

`build`	Build the target in the build root (SYMROOT). This is the default action, and is used if no action is given.
`analyze`	Build and analyze a target or scheme from the build root (SYMROOT). This requires specifying a scheme.
`archive`	Archive a scheme from the build root (SYMROOT). This requires specifying a scheme.
`test`	Test a scheme from the build root (SYMROOT). This requires specifying a scheme and optionally a destination.
`installsrc`	Copy the source of the project to the source root (SRCROOT).
`install`	Build the target and install it into the target's installation directory in the distribution root (DSTROOT).
`clean`	Remove build products and intermediate files from the build root (SYMROOT).

-xcconfig *filename*
Load the build settings defined in *filename* when building all targets. These settings will override all other settings, including settings passed individually on the command line.

`-dry-run, -n`
Print the commands that would be executed, but do not execute them.

`-skipUnavailableActions`
Skip actions that cannot be performed instead of failing. This option is only honored if **`-scheme`** is passed.

`buildsetting`=*`value`*
Set the build setting *`buildsetting`* to *`value`*.

A detailed reference of Xcode build settings can be found at: ⟨**`https://developer.apple.com/documentation/DeveloperTools/Reference/Xcode-BuildSettingRef/`**⟩

-*`userdefault`*=*`value`*
Set the user default *`userdefault`* to *`value`*.

`-toolchain` [*`identifier`* | *`name`*]
Use a given toolchain, specified with either an *`identifier`* or *`name`*.

`-verbose`
Provide additional status output.

`-version`
Display version information for this install of Xcode. Does not initiate a build. When used in conjunction with **`-sdk`**, the version of the specified SDK is displayed, or all SDKs if **`-sdk`** is given no argument. Additionally, a single line of the reported version information may be returned if *`infoitem`* is specified.

`-license`
Show the Xcode and SDK license agreements. Allows for accepting the license agreements without launching Xcode itself, which is useful for headless systems. Must be run as a privileged user.

`-usage`
Displays usage information for **`xcodebuild`**.

Destinations

The **`-destination`** option takes as its argument a *destination specifier* describing the device (or devices) to use as a destination. A destination specifier is a single argument consisting of a set of comma-separated *`key`*=*`value`* pairs. The **`-destination`** option may be specified multiple times to cause **`xcodebuild`** to perform the specified action on multiple destinations.

Destination specifiers may include the *`platform`* key to specify one of the supported destination platforms. There are additional keys which should be supplied depending on the platform of the device you are selecting.

Some devices may take time to look up. The **`-destination-timeout`** option can be used to specify the amount of time to wait before a device is considered unavailable. If unspecified, the default timeout is 30 seconds.

Currently, **`xcodebuild`** supports these platforms:

`OS X` The local Mac, referred to in the Xcode interface as *My Mac*, and which supports the following key:

`arch` The architecture to use, either *`x86_64`* (the default) or *`i386`*.

`iOS` An iOS device, which supports the following keys:

- *`id`* The identifier of the device to use, as shown in the Devices window. A valid destination specifier must provide either *`id`* or *`name`*, but not both.
- *`name`* The name of the device to use. A valid destination specifier must provide either *`id`* or *`name`*, but not both.

`iOS Simulator` A simulated iOS device, which supports the following keys:

- *`id`* The identifier of the simulated device to use, as shown in the Devices window. A valid destination specifier must provide either *`id`* or *`name`*, but not both.
- *`name`* The name of the simulated device to use. A valid destination specifier must provide either *`id`* or *`name`*, but not both.
- *`OS`* When specifying the simulated device by *`name`*, the iOS version for that simulated device, such as *`6.0`*, or the string `latest` (the default) to indicate the most recent version of iOS supported by this version of Xcode.

`watchOS` A watchOS app is always built and deployed nested inside of an iOS app. To use a watchOS device as your destination, specify a scheme which is configured to run a WatchKit app, and specify the iOS platform destination that is paired with the watchOS device you want to use.

`watchOS Simulator` A watchOS Simulator app is always built and deployed nested inside of an iOS Simulator app. To use a watchOS Simulator device as your destination, specify a scheme which is configured to run a WatchKit app, and specify the iOS Simulator platform destination that is paired with the watchOS Simulator device you want to use.

`tvOS` A tvOS device, which supports the following keys:

- *`id`* The identifier of the device to use, as shown in the Devices window. A valid destination specifier must provide either *`id`* or *`name`*, but not both.
- *`name`* The name of the device to use. A valid destination specifier must provide either *`id`* or *`name`*, but not both.

`tvOS Simulator` A simulated tvOS device, which supports the following keys:

- *`id`* The identifier of the simulated device to use, as shown in the Devices window. A valid destination specifier must provide either *`id`* or *`name`*, but not both.
- *`name`* The name of the simulated device to use. A valid destination specifier must provide either *`id`* or *`name`*, but not both.
- *`OS`* When specifying the simulated device by *`name`*, the tvOS version for that simulated device, such as *`9.0`*, or the string `latest` (the default) to indicate the most recent version of tvOS supported by this version of Xcode.

Some actions (such as building) may be performed without an actual device present. To build against a platform generically instead of a specific device, the destination specifier may be prefixed with the optional string "generic/", indicating that the platform should be targeted generically. An example of a generic destination is the "Generic iOS Device" destination displayed in Xcode's UI when no physical iOS device is present.

Exporting Archives

The **-exportArchive** option specifies that **xcodebuild** should export the archive specified by **-archivePath** using the options specified by **-exportOptionsPlist**. **xcodebuild -help** can print the full set of available inputs to **-exportOptionsPlist**. The exported product will be placed at the path specified by **-exportPath**.

Environment Variables

The following environment variables affect the execution of **xcodebuild**:

XCODE_XCCONFIG_FILE
: Set to a path to a file, build settings in that file will be loaded and used when building all targets. These settings will override all other settings, including settings passed individually on the command line, and those in the file passed with the **-xcconfig** option.

Exit Codes

xcodebuild exits with codes defined by sysexits(3). It will exit with **EX_OK** on success. On failure, it will commonly exit with **EX_USAGE** if any options appear malformed, **EX_NOINPUT** if any input files cannot be found, **EX_IOERR** if any files cannot be read or written, and **EX_SOFTWARE** if the commands given to xcodebuild fail. It may exit with other codes in less common scenarios.

EXAMPLES

```
xcodebuild clean install
```

Cleans the build directory; then builds and installs the first target in the Xcode project in the directory from which **xcodebuild** was started.

```
xcodebuild -project MyProject.xcodeproj -target Target1 -target Target2
        -configuration Debug
```

Builds the targets *Target1* and *Target2* in the project *MyProject.xcodeproj* using the *Debug* configuration.

```
xcodebuild -target MyTarget OBJROOT=/Build/MyProj/Obj.root
        SYMROOT=/Build/MyProj/Sym.root
```

Builds the target *MyTarget* in the Xcode project in the directory from which **xcodebuild** was started, putting intermediate files in the directory */Build/MyProj/Obj.root* and the products of the build in the directory */Build/MyProj/Sym.root*.

```
xcodebuild -sdk macosx10.6
```

Builds the Xcode project in the directory from which **xcodebuild** was started against the Mac OS X 10.6 SDK. The canonical names of all available SDKs can be viewed using the **-showsdks** option.

```
xcodebuild -workspace MyWorkspace.xcworkspace -scheme MyScheme
```

Builds the scheme *MyScheme* in the Xcode workspace *MyWorkspace.xcworkspace*.

```
xcodebuild -workspace MyWorkspace.xcworkspace -scheme MyScheme archive
```

Archives the scheme *MyScheme* in the Xcode workspace *MyWorkspace.xcworkspace*.

```
xcodebuild -workspace MyWorkspace.xcworkspace -scheme MyScheme
        -destination 'platform=OS X,arch=x86_64' test
```

Tests the scheme *MyScheme* in the Xcode workspace *MyWorkspace.xcworkspace* using the destination described as *My Mac 64-bit* in Xcode.

```
xcodebuild -workspace MyWorkspace.xcworkspace -scheme MyScheme
        -destination 'platform=iOS Simulator,name=iPhone 5s' -destination
        'platform=iOS,name=My iPad' test
```

Tests the scheme *MyScheme* in the Xcode workspace *MyWorkspace.xcworkspace* using both the iOS Simulator device named *iPhone 5s* for the *latest* version of iOS, and the iOS device named *My iPad*. (Note that the shell requires arguments to be quoted or otherwise escaped if they contain spaces.)

```
xcodebuild -workspace MyWorkspace.xcworkspace -scheme MyScheme
        -destination generic/platform=iOS build
```

Builds the scheme *MyScheme* in the Xcode workspace *MyWorkspace.xcworkspace* using the generic *iOS Device* destination.

```
xcodebuild -exportArchive -archivePath MyMobileApp.xcarchive -exportPath
        ExportDestination -exportOptionsPlist 'export.plist'
```

Exports the archive *MyMobileApp.xcarchive* to the path *ExportDestination* using the options specified in *export.plist*.

```
xcodebuild -exportLocalizations -project MyProject.xcodeproj
        -localizationPath MyDirectory -exportLanguage zh-hans
        -exportLanguage es-MX
```

Exports two XLIFF files to *MyDirectory* from *MyProject.xcodeproj* containing development language strings and translations for Simplified Chinese and Mexican Spanish.

```
xcodebuild -exportLocalizations -project MyProject.xcodeproj
        -localizationPath MyDirectory
```

Export a single XLIFF file to *MyDirectory* from *MyProject.xcodeproj* containing only development language strings. (In this case, the **−exportLanguage** parameter has been excluded.)

```
xcodebuild -importLocalizations -project MyProject.xcodeproj
        -localizationPath MyLocalizations.xliff
```

Imports localizations from *MyLocalizations.xliff* into *MyProject.xcodeproj. Translations with issues will be reported but not imported.*

SEE ALSO

ibtool(1), sysexits(3), xcode-select(1), xcrun(1), xed(1)

Xcode Builds Settings Reference ⟨https://developer.apple.com/documentation/DeveloperTools/Reference/XcodeBuildSettingRef/⟩