Xcode Plugin

Plugin Information
View Xcode integration on the plugin site for more information.

I need your help!!

To help us maintain the plugin, you can take contact to jerome.lacoste *at* gmail.com in particular if you are able to test upcoming releases

Thank you!

A problem, An idea ?

Please use our tasks and issues tracker to report bugs, improvements or new feature.

- Report a bug
- Ask for a new feature
- Ask for an improvement of an existing feature

Also if you want to propose some code change using a Github pull request, please open also a Jira issue. It is easier for developers to track them.

This plugin adds the ability to call Xcode command line tools to automate build and packaging iOS applications (iPhone, iPad, ...).

Table of Contents

Features

Versioning

This builder can invoke agvtool to dynamically update the `CFBundleVersion` (a.k.a. Technical Version) or `CFBundleShortVersionString` (a.k.a. Marketing Version). It supports all macros from the Token Macro Plugin. For example the value `${BUILD_NUMBER}` will be replaced with the current build number. We advice you to generate a unique value for each build for the `CFBundleVersion` if you want for example deploy it into a private store. In that case you can use for example : `${JOB_NAME}-${BUILD_NUMBER}`

Building

The target (optional), configuration (e.g. `Debug`, `Release`) and SDK (optional) can be specified in the per-project config along with whether to perform a clean before the build phase.

Packaging

The builder can be used to package the .app into a .ipa. Since a .app is actually a directory, it can be awkward to work with and, especially, distribute. We can simplify this by packaging the .app into a single .ipa file, which is just a
zip file with a well-known internal structure

Signing

This plugin has two features that help you simplify the code signing process.

Allowing Jenkins to stage developer profile

This plugin builds on top of Credentials Plugin to allow you to store your Apple Developer Profile (*.developerprofile) file. This file contains a code signing private key, corresponding developer/distribution certificates, and mobile provisioning profiles. You can create this file from your XCode.

To upload the developer profile into Jenkins, go to "Manage Credentials" in the system configuration page, and select "Apple Developer Profile" from the "Add" button. See Credentials Plugin for further details.

To use this profile for signing, add "Import developer profile" as a build step before you run xcode, and select the developer profile to import into the build machine. This build step will install the developer profile into the slave's keychain.

Work with existing populated keychains

If you create a dedicated keychain that stores the code signing private key and certificate on every build slave, you can configure the plugin to just unlock this keychain (so that xcode can use it for signing)

> **Unlocking**
>
> If you store your keys in the user keychain (most probably) and if you run jenkins as daemon, make sure you installed a recent version of jenkins for Mac, or configured the daemon to create a session (through the CreationSession=true parameter in your org.jenkins-ci.plist daemon config file). See also

> **Timeouts**
>
> The keychain has a default timeout of 300 seconds. One can override it using the security set-keychain-settings command, which isn't supported in the plugin today. Support in the plugin could be improved by adding options to modify the timeout as part of the build process. See JENKINS-17184.

Unit Testing

This plugin will listen to the xcodebuild output when running OCUnit or XC tests and write out JUnit-format test reports that Jenkins can understand. Jenkins will then use these to publish test failure reports and graphs.

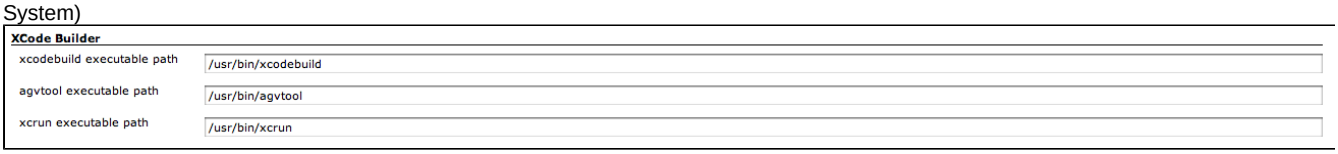
Sample of the configuration panel

Documentation

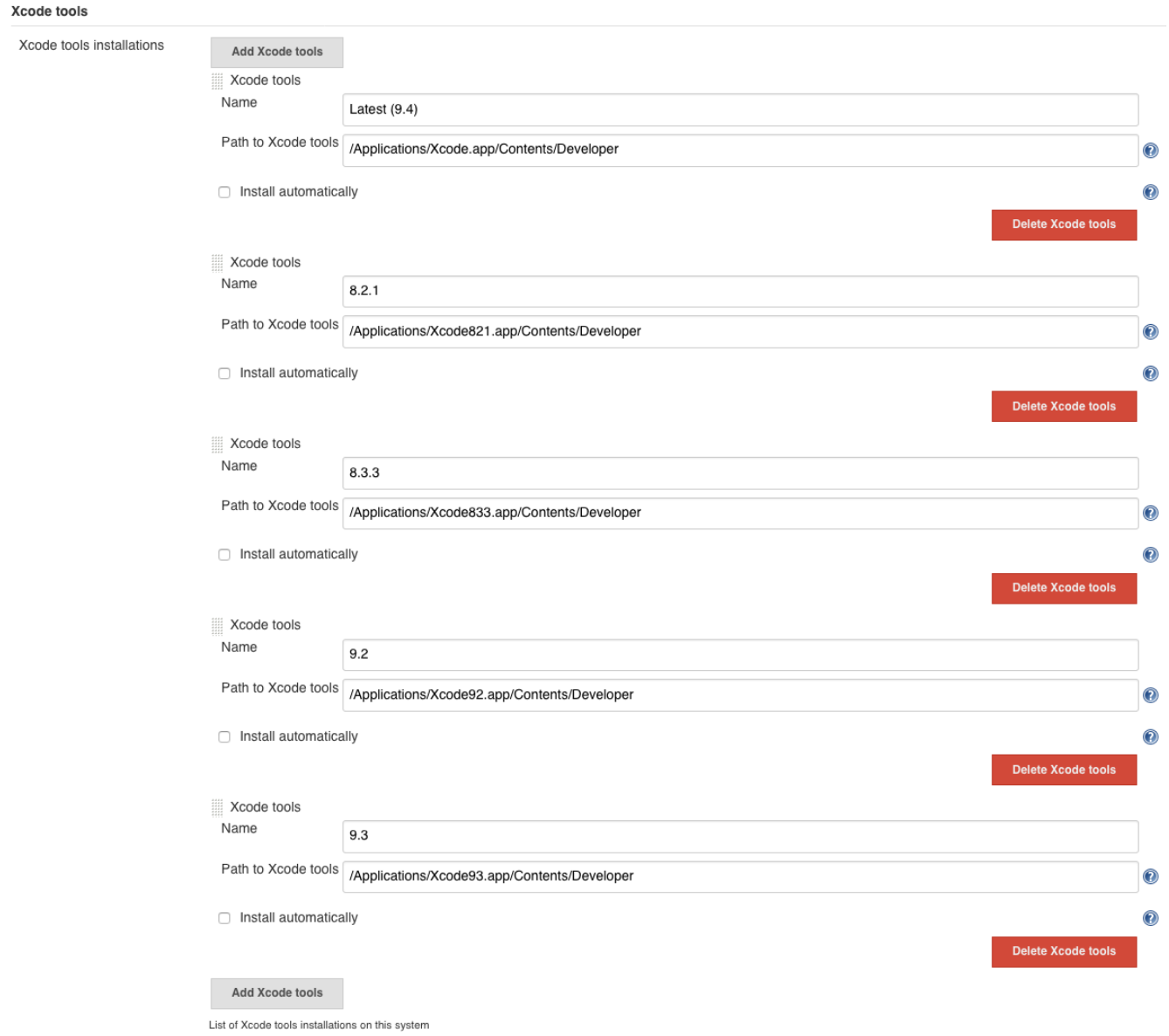
Installation guide

Obviously, the build machine has to be an OSX machine with XCode developer tools installed.
Certificates, Identities and Provisions must be installed on the build machine separately.
Just install the latest version of the plugin from the update center and configure a freestyle job (see #Usage Guide) that will use your OSX node (If your central instance isn't running under OSX).

If xcode related binaries aren't stored in the default location, update the global configuration of the plugin (Manage Jenkins -> Configure System)



If you want to use multiple versions of Xcode to build your application, use 'Global Tool Configuration' to register the distinguished name and location of the installed Xcode.



Usage guide

Import developer profile step

By adding the Import developer profile step to Build, you can import developer profiles exported to Xcode into temporary keychains or existing keychains.
The developer profile contains a set of certificates, secret keys and provisioning profiles necessary for compiling and signing applications, which is useful for distributing with multiple Jenkins nodes.

Import developer profile configuration parameters

Parameter	Pipeline Parameter	Since version	Description

Developer Profile	profileId	2.0.1	Select the developer profile you exported from Xcode Preference to import into the keychain. Developer profile must be registered with Jenkins in advance by "Credentials". If you do not select this column, you can specify the identifier of the developer profile registered in "Credentials" with the next "Developer Profile ID".
Import into existing Keychain	importIntoExistingKeychain	2.0.6	If it is checked, the developer profile will be imported into the existing keychain. If not checked, create a new key chain with the file name "jenkins-" + "job name" newly, and import the developer profile into this. (In this case, a random character string is automatically generated and used for the password)
Target keychain	keychainName	2.0.6	The name of the configured keychain to use to import certificates from developer profile to sign the package.
Keychain path	keychainPath	2.0.6	Keychain path to import developer profile.
Keychain password	keychainPwd	2.0.6	Password to unlock keychain importing developer profile.

Limitations

The Xcode Integration Plugin has the function of importing the secret key and certificate of the developer account into the temporary key chain, but as the constraint of macOS can not handle different key chain for each session, the job handling the key chain If you execute them in parallel, this will not work properly.

As one of workarounds, it is impossible to limit the maximum number of jobs around Jenkins' node to 1 and to execute multiple nodes with different macOS accounts.

Setting up a build step

Add the XCode build step to a free-style project and set the target (e.g. `MyApp`), configuration (e.g. `Release`) and check the build IPA and update version number options. This will give you a single, versioned .ipa file. You can also select to clean the project before a build; while this will make sure that the project is starting from a clean state, it will make large projects take a lot longer.

Setting up a unit test step

Add the XCode build step and this time specify your unit test target (e.g. `MyAppTests`), configuration (e.g. `Debug`) and the SDK (e.g. `Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator4.1.sdk/`).
Leave all the checkboxes unselected. The SDK needs to be supplied here as the unit tests will only be run when building under the simulator's SDK.
Check the "Publish JUnit test result report" option in the project config and set the value "`*/test-reports/*.xml`" under "Test report XMLs". This will tell Jenkins to pick up the JUnit-format test reports.

XCode Builder configuration parameters

Parameter	Pipeline Parameter	Since version	Description
Development Team	developmentTeamName	1.4	The name of the configured development team to use to sign the package. As of XCode 8.3, this is required now to sign an ipa (which now uses xcodebuilder). A development team can be configured here by specifying a Development Team ID, or by creating one in the jenkins global configuration.
Development Team ID	developmentTeamID	1.4	The ID of the Apple development team to use to sign the IPA If 'Development Team' is not selected.

Xcode Tools Version	xcodeName	2.0.3	You can select the version of Xcode set by 'Global Tool Configuration' by plugin setting. If you choose the 'System Default' for Xcode version, plugin will not do anything, so you can switch Xcode using environment variable 'DEVELOPER_DIR' as well.
Use Legacy Build System?	useLegacyBuildSystem	2.0.9	Use the old 'Legacy Build System' instead of 'New Build System' of Xcode 9 and later.
Target	target	1.0	The target to build. If left empty, this will build all targets in the project. If you wish to build your binary and the unit test module, it is best to do this as two separate steps each with their own target. This was, the iPhone Simulator SDK can be specified for the unit tests.
Interpret As Regular Expression	interpretTargetAsRegEx	1.4	Build all entries listed under the "Targets:" section of the xcodebuild -list output that match the regexp.
Clean before build?	cleanBeforeBuild	1.0	This will delete the build directories before invoking the build. This will force the rebuilding of ALL dependencies and can make large projects take a lot longer.
Allow failing build results?	allowFailingBuildResults	1.4	Checking this option will prevent this build step from failing if xcodebuild exits with a non-zero return code.
No Console Log?	noConsoleLog	2.0.1	Checking this option will not log xcode build output to console output.
Logfile Output directory	logfileOutputDirectory	2.0.1	Specify the directory to output the log of xcodebuild. If you leave it blank, it will be output to "project directory/builds /${BUILD_NUMBER}/log" with other logs. If an output path is specified, it is output as a xcodebuild.log file in a relative directory under the "build output directory"
Configuration	configuration	1.0	This is the name of the configuration as defined in the XCode project. By default there are Debug and Release configurations.
Xcode Schema File	xcodeSchema	1.2	Needed if you want to compile for a specific schema instead of a target, or if you want to generate an archive or an IPA.
Generate Archive?	generateArchive	2.0	Checking this option will generate an xcarchive of the specified scheme. A workspace and scheme are are also needed for archives.
Pack application, build and sign . ipa?	buildIpa	1.0	The target to build. If left empty, this will build all targets in the project. If you wish to build your binary and the unit test module, it is best to do this as two separate steps each with their own target. This was, the iPhone Simulator SDK can be specified for the unit tests.
Export method	ipaExportMethod	1.4	The export method of the .app to generate the .ipa file. Should be one in 'development', 'ad-hoc', 'enterprise' or 'app-store'.
.ipa filename pattern	ipaName	1.4	A pattern for the ipa file name. You may use ${VERSION} and ${BUILD_DATE} (yyyy.MM.dd) in this string

Output directory	ipaOutputDirectory	1.4	The output directory for the .ipa file, relative to the build directory.
Upload Bitcode?	uploadBitcode	2.0.3	If Checked, include bitcode in IPA exporting for App Store. The default is "Yes" (check is on).
Upload Symbols?	uploadSymbols	2.0.3	If Checked, include symbols in IPA exporting for App Store. The default is "Yes" (check is on).
Compile Bitcode?	compileBitcode	2.0.3	If checked, Xcode will recompiling Bitcode and exporting IPA for the non App Store. The default is "Yes" (check is on).
Thinning	thinning	2.0.3	Set this to thin the packages for one or more devices in Xcode when exporting IPA by methods other than App Store. Available options are: • <none> (Xcode export non-thinned generic application), • <thin-for-all-variants> (Xcode export universal applications and all available thinning variants), • Or for a model identifier a specific device (eg "iPhone 7, 1"). The default is <none>.
Pack on demand resources?	~~embedOnDemandResourcesAssetPacksInBundle~~ assetPacksInBundle	2.0.3	Export settings for non App Store. If the application is using on demand resources and this item is on, the asset pack is included in the application bundle, so you can test the app without the server hosting the asset pack. If assetPacksBaseURL is not specified, the default is YES (check is on).
On demand resources asset URL	~~onDemandResourcesAssetPacksBaseURL~~ assetPacksBaseURL	2.0.3	Export settings for non-App Store. If the application is using an on demand resource and assetPacksInBundle is not YES, this must be the base URL that specifies the host of the asset pack. This will set up the app to download the asset pack from the specified URL.
Application URL	appURL	2.0.3	Specify the URL for installing the application when creating the manifest.plist that is necessary when OTA installing the application from Web pages or email.
Display image URL	displayImageURL	2.0.3	Export settings for non App Store. Specify the URL of the image (57 x 57 pixels) of the icon to be displayed when installing the application using manifest.plist.
Full size image URL	fullSizeImageURL	2.0.3	Export settings for non App Store. Specify the URL of the image (512 x 512 pixels) of the icon to be displayed when installing the application using manifest.plist.
Asset pack URL	assetPackManifestURL	2.0.3	Export settings for non-App Store App. If the application is using on-demand resources and the application will installing OTA with manifest.plist, this must be the base URL that specifies the host of the asset pack. This will set up the app to download the asset pack from the specified URL.
Strip Swift Symbols?	stripSwiftSymbols	2.0.5	If checked, symbols be stripped from Swift libraries when exporting the application to IPA.
~~Manifest Plist URL~~	~~ipaManifestPlistUrl~~	~~1.5~~	~~The base URL to use to create a Manifest Plist. If omitted no Manifest Plist will be generated.~~

Manual signing?	manualSigning This has been deprecated and has now been replaced by "signingMethod".	2.0.1	For this option you need to manually specify the combination of Provisioning profile UUID and BundleID. This will be useful if you want to change the combination of Provisioning Profile and Certificate used for CodeSign when you build the application.
Code signing settings	signingMethod	2.0.7	This attribute can take one of the following three values. **"aurtomatic"** (Check "Automatic Signing") Checking this option will automatically generate Provisioning Profile and certificates for signing application. However, please be aware that using this function will automatically create Provisioning profile and certificates as necessary, so that old Provisioning profile and certificates will be invalid at that time. **"manual"** (Check "Manual signing") For this option you need to manually specify the combination of Provisioning profile UUID and BundleID. This will be useful if you want to change the combination of Provisioning Profile and Certificate used for CodeSign when you build the application. **"readFromProject"** (Check "Read from Xcode Project") With this option, it automatically retrieve and sets the combination of BundleID and Provisioning profile UUID from the Xcode project file. This is useful when you want to take over the combination of BundleID and Provisioning profile UUID that you used in the Xcode project (GUI).
Provisioning Profiles Bundle ID	provisioningProfiles: [provisioningProfileAppId:]	2.0.1	Specify the Bundle ID of the application for which code sign to be performed. If the location of the Info.plist file contained in the compiled archive is set instead of the Bundle ID, read the Bundle ID from the Info. plist file and use that value.
Provisioning profiles UUID	provisioningProfiles: [provisioningProfileUUID]	2.0.1	Specify the UUID or Specifire of the provisioning profile to use to sign the application. If the location of the provisioning profile is set instead of the UUID or Specifire, read the UUID from the provisioning profile and use that value.

Copy provisioning profile	copyProvisioningProfile	2.0.7	If Checked, and the filename of provisioning profile is specified in "Provisioning profile UUID", the specified file is copied to a predetermined place. ("/Users /${HOME}/Library/MobileDevice /Provisioning Profiles/") This can be used to overwrite the contents of the developer profile or system settings in the provisioning profile in the project deployed from SCM etc. If the provisioning profile UUID or Specifire is set in "provisioning profile UUID", it does not do anything. The default is "Yes" (check is on)
Change bundle ID?	changeBundleID	1.4	Checking this option will replace the bundle identifier. You will need to specify which bundle ID (CFBundleIdentifier) to use and where is the Info.plist file located. This is handy for example when you want to use a different code signing identity in your development projects.
New bundle ID	bundleID	1.4	The new bundle ID. Usually something like com.companyname. projectname.
Info.plist path	bundleIDInfoPlistPath	1.4	The path to the info.plist file which contains the CFBundleIdentifier of your project. Usually something like: ${WORKSPACE}/ProjectName /Project-Info.plist
Unlock Keychain?	unlockKeychain	1.0	Automatically unlock the keychain before signing the archive?
Keychain name	keychainName	1.4	The name of the configured keychain to use to retrieve certificates to sign the package.
Keychain path	keychainPath	1.0	The path of the keychain to use to retrieve certificates to sign the package (default : `${HOME} /Library/Keychains/login. keychain`).
Keychain password	keychainPwd	1.0	The password of the keychain to use to retrieve certificates to sign the package.
Clean test reports?	cleanTestReports	1.3	This will delete the processed test reports before invoking the build. Usually it is a good idea only to do it in test targets, otherwise if other Xcode target is built before tests – the build will fail to collect test reports.
SDK	sdk	1.0	You only need to supply this value if you want to specify the SDK to build against. If empty, the SDK will be determined by XCode. If you wish to run OCUnit tests, you will need to use the iPhone Simulator's SDK, for example: `/Developer /Platforms /iPhoneSimulator.platform /Developer/SDKs /iPhoneSimulator4.1.sdk/`

SYMROOT	symRoot	1.1	You only need to supply this value if you want to specify the SYMROOT path to use. If empty, the default SYMROOT path will be used (it could be different depending of your Xcode version). Supports all macros and also environment and build variables from the Token Macro Plugin. For example you can use the value : `${WORKSPACE}/build`
Custom xcodebuild arguments	xcodebuildArguments	1.3	Passing custom arguments is convenient when you need to change some project options for Jenkins build, but don't want them to persist in project file. For example when running application tests, following settings often are nice fit: `GCC_SYMBOLS_PRIVATE_EXTERN=NO COPY_PHASE_STRIP=NO`. Otherwise Release build will usually fail because it won't be able to find application symbols to link with test bundle. Arguments are currently separated by space, enclosing them in quotes won't help.
Xcode Workspace File	xcodeWorkspaceFile	1.2	Only needed if you want to compile a workspace instead of a project. It takes precedence over 'Xcode Project File' setting and this job 'configuration' parameter."
Xcode Project Directory	xcodeProjectPath	1.0	This is the relative path from the workspace to the directory that contains the XCode project file. You only need to supply this value if the XCode project you wish to build is not in the root of the workspace.
Xcode Project File	xcodeProjectFile	1.0	If there is more than one XCode project file in the project path, you will need to specify the file name of the project you wish to build. If you need to build all project, you will need to create an XCode build step for each one manually.
Build output directory	buildDir	1.2	The value to use for CONFIGURATION_BUILD_DIR setting (BUILD_DIR in >= 2.0.0). You only need to supply this value if you want the product of the XCode build to be in a location other than the one specified in project settings and this job 'SYMROOT' parameter. Supports all macros and also environment and build variables from the Token Macro Plugin. For example you can use the value : `${WORKSPACE}/build`
Provide version number and run avgtool?	provideApplicationVersion	1.4	Provide version number and run avgtool before invoking the build.
Marketing version	cfBundleShortVersionStringValue	1.0	This will set the `CFBundleShortVersionString` to the specified string. Supports all macros and also environment and build variables from the Token Macro Plugin.

Technical version	cfBundleVersionValue	1.0	This will set the `CFBundleVersion` to the specified string. Supports all macros and also environment and build variables from the Token Macro Plugin. For example the value `${BUILD_NUMBER}` will be replaced with the current build number. We advice you to generate a unique value for each build if you want for example deploy it into a private store. In that case you can use for example : `${JOB_NAME}-${BUILD_NUMBER}`

Setting up a Export IPA step

If you add the Export IPA step and set up the information necessary for exporting the IPA file, the location of the archive, the export method, code signature information, etc., you can export the IPA from the archive output in the Xcode Build step.

Export IPA configuration parameters

Parameter	Pipeline Parameter	Since version	Description
Development Team	developmentTeamName	2.0.1	The name of the configured development team to use to sign the package. As of XCode 8.3, this is required now to sign an ipa (which now uses xcodebuilder). A development team can be configured here by specifying a Development Team ID, or by creating one in the jenkins global configuration.
Development Team ID	developmentTeamID	2.0.1	The ID of the Apple development team to use to sign the IPA If 'Development Team' is not selected.
Xcode Tools Version	xcodeName	2.0.3	You can select the version of Xcode set by 'Global Tool Configuration' by plugin setting. If you choose the 'System Default' for Xcode version, plugin will not do anything, so you can switch Xcode using environment variable 'DEVELOPER_DIR' as well.
Archive directory	archiveDir	2.0.1	Specify the location of the path (usually BUILD_DIR specified by xcodebuild) to read the Archive for exporting the IPA file. Supports all macros and also environment and build variables from the Token Macro Plugin. For example you can use the value : ${WORKSPACE}/build
Export method	ipaExportMethod	2.0.1	The export method of the .app to generate the .ipa file. Should be one in 'development', 'ad-hoc', 'enterprise' or 'app-store'.
.ipa filename pattern	ipaName	2.0.1	A pattern for the ipa file name. You may use ${VERSION} and ${BUILD_DATE} (yyyy.MM.dd) in this string
Output directory	ipaOutputDirectory	2.0.1	The output directory for the .ipa file, relative to the build directory.
Upload Bitcode?	uploadBitcode	2.0.3	If Checked, include bitcode in IPA exporting for App Store. The default is "Yes" (check is on).
Upload Symbols?	uploadSymbols	2.0.3	If Checked, include symbols in IPA exporting for App Store. The default is "Yes" (check is on).

Compile Bitcode?	compileBitcode	2.0.3	If checked, Xcode will recompiling Bitcode and exporting IPA for the non App Store. The default is "Yes" (check is on).
Thinning	thinning	2.0.3	Set this to thin the packages for one or more devices in Xcode when exporting IPA by methods other than App Store. Available options are: • <none> (Xcode export non-thinned generic application), • <thin-for-all-variants> (Xcode export universal applications and all available thinning variants), • Or for a model identifier a specific device (eg "iPhone 7, 1"). The default is <none>.
Pack on demand resources?	assetPacksInBundle	2.0.3	Export settings for non App Store. If the application is using on demand resources and this item is on, the asset pack is included in the application bundle, so you can test the app without the server hosting the asset pack. If assetPacksBaseURL is not specified, the default is YES (check is on).
On demand resources asset URL	assetPacksBaseURL	2.0.3	Export settings for non-App Store. If the application is using an on demand resource and assetPacksInBundle is not YES, this must be the base URL that specifies the host of the asset pack. This will set up the app to download the asset pack from the specified URL.
Application URL	appURL	2.0.3	Specify the URL for installing the application when creating the manifest.plist that is necessary when OTA installing the application from Web pages or email.
Display image URL	displayImageURL	2.0.3	Export settings for non App Store. Specify the URL of the image (57 x 57 pixels) of the icon to be displayed when installing the application using manifest.plist.
Full size image URL	fullSizeImageURL	2.0.3	Export settings for non App Store. Specify the URL of the image (512 x 512 pixels) of the icon to be displayed when installing the application using manifest.plist.
Asset pack URL	assetPackManifestURL	2.0.3	Export settings for non-App Store App. If the application is using on-demand resources and the application will installing OTA with manifest.plist, this must be the base URL that specifies the host of the asset pack. This will set up the app to download the asset pack from the specified URL.
Strip Swift Symbols?	stripSwiftSymbols	2.0.5	If checked, symbols be stripped from Swift libraries when exporting the application to IPA.
Manual signing?	manualSigning This has been deprecated and has now been replaced by "signingMethod".	2.0.1	For this option you need to manually specify the combination of Provisioning profile UUID and BundleID. This will be useful if you want to change the combination of Provisioning Profile and Certificate used for CodeSign when you build the application.

Code signing settings	signingMethod	2.0.7	This attribute can take one of the following three values. **"aurtomatic"** (Check "Automatic Signing") Checking this option will automatically generate Provisioning Profile and certificates for signing application. However, please be aware that using this function will automatically create Provisioning profile and certificates as necessary, so that old Provisioning profile and certificates will be invalid at that time. **"manual"** (Check "Manual signing") For this option you need to manually specify the combination of Provisioning profile UUID and BundleID. This will be useful if you want to change the combination of Provisioning Profile and Certificate used for CodeSign when you build the application. **"readFromProject"** (Check "Read from Xcode Project") With this option, it automatically retrieve and sets the combination of BundleID and Provisioning profile UUID from the Xcode project file. This is useful when you want to take over the combination of BundleID and Provisioning profile UUID that you used in the Xcode project (GUI).
Provisioning Profiles Bundle ID	provisioningProfiles: [provisioningProfileAppId:]	2.0.1	Specify the Bundle ID of the application for which code sign to be performed. If the location of the Info.plist file contained in the compiled archive is set instead of the Bundle ID, read the Bundle ID from the Info. plist file and use that value.
Provisioning profiles UUID	provisioningProfiles: [provisioningProfileUUID]	2.0.1	Specify the UUID or Specifire of the provisioning profile to use to sign the application. If the location of the provisioning profile is set instead of the UUID or Specifire, read the UUID from the provisioning profile and use that value.
Unlock Keychain?	unlockKeychain	2.0.1	Automatically unlock the keychain before signing the archive?
Copy provisioning profile	copyProvisioningProfile	2.0.7	If Checked, and the filename of provisioning profile is specified in "Provisioning profile UUID", the specified file is copied to a predetermined place. ("/Users /${HOME}/Library/MobileDevice /Provisioning Profiles/") This can be used to overwrite the contents of the developer profile or system settings in the provisioning profile in the project deployed from SCM etc. If the provisioning profile UUID or Specifire is set in "provisioning profile UUID", it does not do anything. The default is "Yes" (check is on)

Keychain name	keychainName	2.0.1	The name of the configured keychain to use to retrieve certificates to sign the package.
Keychain path	keychainPath	2.0.1	The path of the keychain to use to retrieve certificates to sign the package (default : `${HOME}/Library/Keychains/login.keychain`).
Keychain password	keychainPwd	2.0.1	The password of the keychain to use to retrieve certificates to sign the package.

Using multiple versions of xcode.

One can easily support multiple versions by using environment variables. But first one need to install multiple versions on the build server. Here's one strategy:

install multiple versions of xcode

0. Note current config.

xcode-select -p

1. download xcode DMGs from http://developer.apple.com/downloads/

2. enable install from everywhere (under System preferences / Security)

Without that I had problems installing xcode, without graphical feedback, tested on 10.9.5, freshly booted.

Symptoms were:

- app not responsive
- Getting errors like in /var/log/system.log

Someone attempted to start application App:"Xcode" asn:0x0-1b62b61 pid:57656 refs=6 @ 0x7fa5d9f6df40 but it still has _kLSApplicationLockedInStoppedStateKey=true, so it is is staying stopped. : LASApplication.cp #2468 SetApplicationInStoppedState() q=LSSession 100027/0x186bb queue

3. open DMG file, copy app to /Applications. This might take a while.

4. close /Volumes/Xcode (otherwise installation won't start)

5. start Xcode, accept agreement, install missing packages if necessary. This might take a while. Close xcode after GUI opens.

6. move freshly installed Xcode to new path, e.g.

sudo mv /Applications/Xcode.app /Applications/Xcode6.1.app

(I use this so that the xcode app appears with version number under spotlight)

7. reset default xcode-select if necessary (installation might have changed it / reset to /Applications/Xcode.app)

sudo xcode-select -s /Applications/Xcode6.1.app

(Note the part above could be fully automated....)

Select the xcode version at runtime

1. Use EnvInject plugin
2. for jobs that require the non default do something like

DEVELOPER_DIR=/Applications/Xcode6.0.1.app/Contents/Developer

under Inject environment variables to the build process

Select the xcode version at runtime (Version 2.0.3 or later)

1. Register the distinguished name and location of the installed Xcode using 'Global Tool Configuration'.
2. Select Xcode registered in 'Global Tool Configuration' in Job's 'Xcode Tools Version'.

That's it.

FAQ

User interaction is not allowed

When code-signing, a prompt may appear on the build machine asking whether to allow keychain access.
This will block the build with a message like `"User interaction is not allowed"` until it is dismissed.
Just select 'Always Allow' the first time and it shouldn't need to ask again.

If this prompt is not showing on the build machine, you can force it to appear by running the codesign command that failed from a terminal on the build machine: /usr/bin/codesign --force --sign "iPhone Distribution: .....

No Xcode folder is set

```
[sources] $ /usr/bin/xcodebuild -version
xcode-select: Error: No Xcode folder is set. Run xcode-select -switch
<xcode_folder_path> to set the path to the Xcode folder.
Error: /usr/bin/xcode-select returned unexpected error.
FATAL: Check your XCode installation. Jenkins cannot retrieve its
version.
```

You probably just installed XCode (by default it is in `/Developer`) but its path isn't configured. You can fix it with

```
xcode-select -switch /Developer/
```

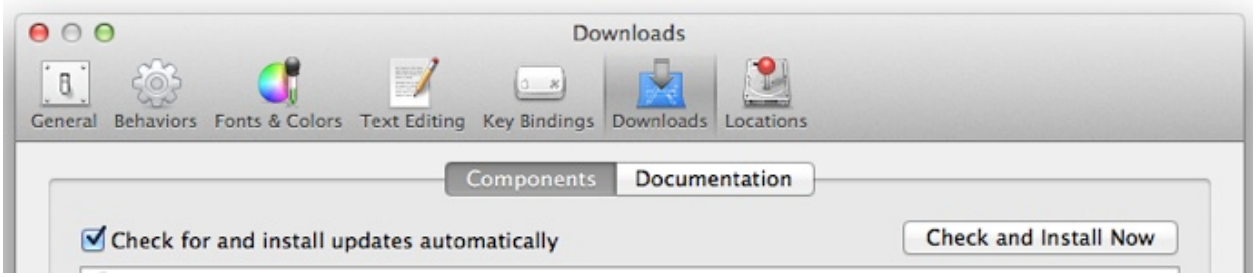
You should now be able to execute `xcodebuild`. Ex :

```
host:~ user$ /usr/bin/xcodebuild -version
Xcode 4.2
Build version 4C199
```

Xcode 4.3

XCode 4.3 changes 2 things that are important for our usage on a slave or a master :

- **Command-Line Tools Are Optional** : The command-line tools are not bundled with Xcode 4.3 by default. Instead, they have to be installed using the Components tab of the Downloads preferences panel



- **/Developer No Longer Exists** : You may have to reconfigure its path like this:

```
sudo xcode-select -switch /Applications/Xcode.app/Contents
/Developer/
```

Changelog

Version 2.0.10 (19th Nov 2018)

- **Fix broken Xcode Project Parser. (JENKINS-54414, JENKINS-54113)**

Version 2.0.9 (2nd Nov 2018)

- **Added a option to use the 'Legacy Build System' instead of 'New Builld System' which became available from Xcode 9.**

Version 2.0.8 (10th Oct 2018)

- Corresponds when there is no DEVELOPMENT_TEAM entry in the old Xcode project.
- Delete unnecessary error messages.
- When copying the provisioning profile from the project location, it needs to be done before Xcode compilation, so it is fixed.
- Fixed a bug that error message got mixed when parsing provisioning profile and analysis failed.

Version 2.0.7 (20th Sep 2018)

- Added the ability to copy provisioning profile files to manual code signing.
- Change the help for importing developer profile to more detailed one.
- Enhance backward compatibility.
- Added a function to retrieve information necessary for CodeSign from Xcode project.

Version 2.0.6 (16th Aug 2018)

- Fixed a bug that key chain was not unlocked properly when importing developer profile to keychain.
- Added input validation when setting up developer profile loader.
- Added input validation when setting up export IPA.
- Fix some bugs.

Version 2.0.5 (9th Aug 2018)

- More compatibility for Pipeline.
- Added function to import developer profile into existing keychain.
- Added a function to set exportOptions.plist an option for deleting Swift symbols when exporting IPA.

Version 2.0.4 (22th Jun 2018)

- I will display an error message if the developer profile is not loaded.

Version 2.0.3 (18th Jun 2018)

- Implemented the ability to export IPA files from already compiled archives.
- JENKINS-50266 : Fix custom xcodebuild arguments not passed through to export archive step.
- JENKINS-51418 : Fix to always be able to use developer team ID as a parameter regardless of automatic code signature.
- Fix the Plugin is abnormally terminate if the out of order of lines from Xcodebuild output is changed.
- Added a function to retrieve necessary information for code signature from compiled archive.
- Support multiple versions of Xcode without using 'EnvInject Plugin'.

Version 2.0.2 (30th Apr 2018)

- We released again due to infrastructure related problems, but the contents are the same as Ver 2.0.1

Version 2.0.1 (26th Apr 2018)

- Supports Xcode 9 (PR #86, PR#87, JENKINS-47744, JENKINS-45509)
- Adding Pipeline support for importDeveloperProfile and adding symbols (PR #89)

Version 2.0.0 (24th May 2017)

- The plugin now requires Java 7 and Jenkins core >= 1.625.1

- The plugin now requires Xcode >= 7
- New : Supports Xcode 7/8 (PR#76, PR#78, JENKINS-44203, JENKINS-44151, JENKINS-43226, JENKINS-43163, JENKINS-38799, JENKINS-38777, JENKINS-34307)
    - Replace 'xcrun PackageApplication' by 'xcodebuild -exportArchive' to package ipa
    - New Development Team parameter which can be configured globally in Jenkins Global settings or locally at the project level
- New : Adds regex for parsing UI test failures (for Xcode 8) (PR#75, JENKINS-40938)
- New : Adds Pipeline Plugin support (PR#74, JENKINS-42457, JENKINS-33355)

Version 1.4.11 (21th September 2016)

Because XXXX happens

- Fix: Support configurable ID for developer profile (JENKINS-32987)

Version 1.4.10 (20th September 2016)

- New: Support configurable ID for developer profile (JENKINS-32987)
- New: Allow the option not to sign the IPA with xcrun (JENKINS-32370)
- Fix: use textarea for Xcode Build Arguments (JENKINS-30228)
- Fix: Generating an archive builds the project twice (JENKINS-30362)
- Fix: Allowed the use of environment variables in plist url (JENKINS-27236)

Version 1.4.9 (23rd September 2015)

- Fix: only prepend -allTargets for -project builds (JENKINS-28256)
- New: Add support for Xcode 7 date format in Unit Testing build task (pull request xcode-plugin/63)

Version 1.4.8 (28th February, 2015)

- Fix: Add timeout to xcodebuild -list to avoid hang on xcode 6 (pull request xcode-plugin/58)
- New: Support multiple targets to be passed into xcodebuild (pull request xcode-plugin/43)

Version 1.4.7 (5th January, 2015)

- Fix: XCTest output parsing: support nested/namespaced test suites (JENKINS-26295)
- New: XCTest output parsing: handle tests suites exiting with an error (pull request xcode-plugin/54)

Version 1.4.6 (16th December, 2014)

- Fix: broken dSYM packaging (since 1.4.5)
- Fix: BUILD_DATE does not produce the correct last modified date if built on a slave machine (pull request xcode-plugin/50)
- Fix: make sure to set provideApplicationVersion properly upon upgrade from per-1.4.1 (JENKINS-26027)
- New: developerProfileLoader: use show-keychain-info to display job specific keychain information
- New: support XC test output (JENKINS-19955)
- New: display warning when simulator SDK selected and IPA about to be packaged (JENKINS-21293)
- New: document xcodebuildArguments (JENKINS-13930)

Version 1.4.5 (10th December, 2014)

- Fix: only zip the DSYM if the DSYM file was actually created and fail the build if the operation failed
- Fix: Fail the build if we fail to create an ipa.
- Fix: "keychain with the same name already exists" (JENKINS-22130)
- New: XCode 6 compatibility (pull request xcode-plugin/48)
- New: generate_manifest added ability to generate enterprise distribution manifest plist (pull request xcode-plugin/45)
- Fix: getKeychain returns a global keychain preferred over path (pull request xcode-plugin/41)
- New: Ability to change the Bundle identifier (CFBundleIdentifier) for an xcode build (pull request xcode-plugin/39)

Version 1.4.2 (31st December, 2013)

- Fix: Avoid NPE on plugin version update (JENKINS-19886, pull request xcode-plugin/37)

Version 1.4.1 (30th October, 2013)

- Fix: performance regression in output parsing (JENKINS-20037, pull request xcode-plugin/36)

Version 1.4 (2nd Oct, 2013)

- New: added an option to produce .xcarchive (JENKINS-14719)
- Fix: Fixing show-keychain-info call (JENKINS-xcode-plugin/30)
- New: Build wrapper to restore the keychain at the end of the build (pull request xcode-plugin/31)
- New: Adding the possibility to provide a pattern for the .ipa file name (pull request xcode-plugin/33)
- New: Added an ability to import *.developerprofile into a build (pull request xcode-plugin/34)
- New: UI update: group options into 3 categories (pull request xcode-plugin/28)
- New: Introduced the ability to create global keychain configurations

- New: Add new option "allow failing build results" (pull request xcode-plugin/25)
- Fix: xcodebuild output parser is more lenient (pull request xcode-plugin/19)
- Fix: Can't add xcodebuild parameters including whitespace (JENKINS-12800)
- New: Add a field for entering code signing identity (pull request xcode-plugin/6)

Version 1.3.1 (27th March 2012)

- Fix: Custom xcodebuild arguments values are not persisted (JENKINS-12510).

Version 1.3 (20th January 2012)

- New : Ability to specify custom arguments to xcodebuild (so that values in project file can be overridden). It is needed to specify custom build options to make in-app unit tests work.
- New : Ability to disable clean up of test reports on per-target level, so that it is possible to run several targets in single job and not mess-up test reports.
- Fix : The plugin fails by searching for a double .app extension while compressing .dsym (JENKINS-12273)
- Fix : The plugin fails to delete previous generated IPA results in a failed build (JENKINS-12237).
- Fix : The plugin fails to set default keychain when using an alternate (non login) keychain (By default the login keychain is the default keychain) (JENKINS-12217).
- Fix : Restore Java 1.5 runtime compatibility. (JENKINS-12378)

Version 1.2.2 (19th December 2011)

- Fix : Build IPA fails if the plugin is launched on a remote agent using a relative path for its FS Root directory (JENKINS-12144)

Version 1.2.1 (18th December 2011)

- Fix : Build IPA switch doesn't work properly for a default Xcode project always results in FATAL error (JENKINS-12089)

Version 1.2 (8th December 2011)

- New build parameter to specify the build output directory. This overrides the setting in the user's project file, so it will be put into a known directory rather than XcodeDerivedData, etc. This makes it much easier to set up subsequent build steps in Jenkins that do something with the output of the build. It does this by passing the setting CONFIGURATION_BUILD_DIR to xcodebuild if a path is set for this new job config value.
- Add support for building schemes and workspaces : The plugin now supports two extra configuration parameters `xcodeScheme` and `xcodeWorkspaceFile`. The scheme maps to xcodebuild's parameter "-scheme" and the workspace to "-workspace". The scheme takes precedence over the target setting and the workspace takes precedence over the project setting.

Version 1.1 (29th November 2011)

- Upgrade the token macro plugin to version 1.5.1 to support environment and build variables (JENKINS-11892)
- New configuration parameter to set the SYMROOT value passed to the build. This parameter accepts environment and build variables and Macros (JENKINS-11813)
- Various code cleanup and improvements.

Version 1.0.1 (14th November 2011)

- Minor fix about the default value and the documentation of the keystore path parameter.

Version 1.0 (14th November 2011)

- Initiated from Ray Yamamoto Hilton's Xcode plugin for Hudson with few changes :
  - It improves jenkins compatibility, and is available in its update center.
  - It uses the Token Macro Plugin to configure values of `CFBundleVersion` and `CFBundleShortVersionString`
  - It allows to configure the keychain to use and to unlock it
  - It is ready for internationalization

> If you previously used the version provided by Ray you should be able to test this one in parallel as it has a new identity. Also you'll have to reconfigure a large part of it if you want to upgrade thus don't forget to save your settings.

Known issues

T	Key	Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated	Due
	JENKINS-55366	I have build the .ipa file in mac i have issue	Unassigned	Senthil Kumar		Open	Unresolved	Dec 31, 2018	Dec 31, 2018	
	JENKIN	NPE at au.com.rayh.BuildConfiguration.<init>	Unassi	Alexey		Ope	Unreso	Dec	Dec	

	S-55207	(BuildConfiguration.java:47)	gned	Molchanov		n	lved	14, 2018	14, 2018
	JENKINS-54973	Not able to generate build With Xcode 10	Unassigned	Ramakrishna Talupula		Open	Unresolved	Dec 03, 2018	Dec 07, 2018
	JENKINS-53966	Import Developer Profile in to keychain.	Unassigned	Kazuhide Takahashi		Open	Unresolved	Oct 10, 2018	Oct 10, 2018
	JENKINS-53663	Xcode 10 parallel builds are not supported	Unassigned	Claus Joergensen		Open	Unresolved	Sep 19, 2018	Sep 21, 2018
	JENKINS-53647	Xcode 10 Support for iOS Build Teams	Unassigned	Dalton Robertson		Open	Unresolved	Sep 18, 2018	Sep 18, 2018
	JENKINS-52770	started.FATAL: Log statements out of sync: current test suite 'Target_Test.xctest' not exists	Unassigned	Mohamed Abdelli		Open	Unresolved	Jul 27, 2018	Aug 05, 2018
	JENKINS-52741	have to supply all arguments when calling from pipeline	Unassigned	Adrian Bridgett		Open	Unresolved	Jul 25, 2018	Aug 06, 2018
	JENKINS-51850	Functions of xcode plugin "Change Bundle ID" does not work properly.	Unassigned	Kazuhide Takahashi		In Progress	Unresolved	Jun 11, 2018	Jun 14, 2018
	JENKINS-51177	For Xcode9 and above,not able to export the archive	Unassigned	Aparna Konakanchi		Open	Unresolved	May 07, 2018	May 07, 2018
	JENKINS-48606	dSYM being archived from the wrong directoty	Unassigned	Deepak Shenoy		Open	Unresolved	Dec 18, 2017	Dec 18, 2017
	JENKINS-48590	Failed to export	Unassigned	km ln		Open	Unresolved	Dec 16, 2017	Dec 16, 2017
	JENKINS-47744	xcodebuild export step fails due to empty exportPlist	Unassigned	Juan Moya		Open	Unresolved	Oct 31, 2017	Mar 06, 2018
	JENKINS-46623	evey build ending with ARCHIVE FAILED	Unassigned	Tomasz Olszewski		Open	Unresolved	Sep 03, 2017	Sep 03, 2017
	JENKINS-46495	You have to provide a value for either the marketing or technical version. Found neither.	li yong	li yong		Open	Unresolved	Aug 28, 2017	Dec 14, 2018
	JENKINS-45509	Xcode 9 requires provisioning profile to be specified when using manual code signing	nguyen hoang	James Addyman		Open	Unresolved	Jul 13, 2017	May 18, 2018
	JENKINS-45363	Build failed: No iOS profile matching 'xxx/xxx_dist' found:	Unassigned	Jogendar Singh		Open	Unresolved	Jul 07, 2017	Jul 07, 2017
	JENKINS-45307	Created Manifest Plist file does not contain bundle identifier and display name	Unassigned	Petra A		Open	Unresolved	Jul 05, 2017	Sep 22, 2017
	JENKINS-45190	Xcode plugin 2.0.0 no longer zips dSYMs as in previous versions	Unassigned	Jordan Hesse		Open	Unresolved	Jun 28, 2017	Apr 20, 2018
	JENKINS-44744	Cannot add a build step from XCode plugin 2.0.0 on IE 11	Unassigned	Arnaud Héritier		Open	Unresolved	Jun 07, 2017	Jun 07, 2017

Showing 20 out of 106 issues