SDKs

Xcode uses SDK bundles to gather the resources needed to build code against a desired target. SDK bundles are comprised of a plist file named `SDKSettings.plist` which dictates the settings used for that particular SDK and how it should be consumed by Xcode. There are two types of SDK bundles, Base and Sparse, which perform different functions. This post documents how to create your own SDK bundles that can be used with Xcode.

Table of Contents

SDK Types

#####Base SDK These are the type of SDKs that ship with Xcode. They contain the all the headers and system libraries/frameworks necessary to build code against a specific OS version. Only one Base SDK can be used for a target for compilation. The Base SDK is assigned by specifying `SDKROOT` for the target.

#####Sparse SDK These are SDK bundles that contain supplimental libraries/ frameworks and headers. Sparse SDKs have been supported for a long time in Xcode. While only one Base SDK can be used, multiple Sparse SDKs can be used in conjunction with the Base SDK to provide access to additional resources.

↑ Table of Contents

Configuration

#####SDKSettings.plist The `SDKSettings.plist` file is a plist that contains information necessary for loading and resolving the SDK within Xcode. This file lives at the root of the SDK bundle.

Key Name	Type	Description
CanonicalName	String	Name of the SDK when being specified to th toolchain
CustomProperties	Dictionary	Custom build settings variables that should imported when using this SDK
DefaultProperties	Dictionary	Default values for build setting variables wh using this SDK
DisplayName	String	The name of the SDK as displayed by Xcode
MinimalDisplayName	String	Alternative (reduced) name for this SDK
DefaultDeploymentTarget	String	Default deployment target of the OS version when using this SDK
MaximumDeploymentTarget	String	Maximum deployment target of the OS versi when using this SDK
MinimumSupportedToolsVersion	String	Lowest supported version of Xcode that this should be used with
SupportedBuildToolComponents	Array [Strings]	Used in SDKs for OS X, contains the string `com.apple.compilers.gcc.headers.4_2`
Version	String	Version of the OS for this SDK
isBaseSDK	String	"YES" or "NO" indicating if this should be tre as a Base SDK
DocSetFeedName	String	Display name of the docset feed for this SDK
DocSetFeedURL	String	URL of the docset feed for this SDK
Toolchains	Array [Strings]	List of identifiers of toolchains that should b used with this SDK
PropertyConditionFallbackNames	Array [Strings]	Used in WatchOS SDK, contains the string `embedded`
AlternateSDK	String	CanonicalName of another SDK to use when targeting this SDK isn't appropriate

#####Custom Environments

Note: This only applies to Base SDKs, Sparse SDKs don't have these settings read from them.

The `DefaultProperties` and `CustomProperties` dictionary items on this plist can be used to enhance the build process. They are treated as SDK level build settings that all targets should inherit. For example, if you create a SDK that must always have `-ObjC` passed, you can add a key to `DefaultProperities` for `OTHER_LDFLAGS` with that flag as the value. This means you won't have to add that flag on the project or target level, it will already be inherited from the SDK. Similiarly you can use `OTHER_CFLAGS` to provide additional library and header search paths for your SDK.

If you need to specify supplimentary variables to use with part of your build process, eg relative paths to a framework or library, they can be provided in the `CustomProperties` dictionary. To create relative paths for any files in your SDK, use `$(SDK_DIR)` as the start of your path. This will provide the path to the root of the SDK bundle. Example of defining a path to a specific framework in the SDK:

```
...
<key>CustomProperties</key>
<dict>
    <key>MY_CUSTOM_FRAMEWORK_NAME</key>
    <string>Foo</string>
    <key>MY_CUSTOM_FRAMEWORK_PATH</key>
    <string>$(SDK_DIR)/System/Library/Frameworks/$(MY_CUSTOM_FRAMEWORK)
</dict>
...
```

#####Example Base SDKSettings.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.
<plist version="1.0">
<dict>
    <key>CanonicalName</key>
    <string>macosx10.9</string>
    <key>CustomProperties</key>
    <dict>
        <key>KERNEL_EXTENSION_HEADER_SEARCH_PATHS</key>
        <string>$(KERNEL_FRAMEWORK)/PrivateHeaders $(KERNEL_FRAMEWORK_H
    </dict>
    <key>DefaultProperties</key>
    <dict>
        <key>MACOSX_DEPLOYMENT_TARGET</key>
        <string>10.9</string>
        <key>PLATFORM_NAME</key>
        <string>macosx</string>
        <key>DEFAULT_KEXT_INSTALL_PATH</key>
        <string>$(LIBRARY_KEXT_INSTALL_PATH)</string>
    </dict>
    <key>DisplayName</key>
    <string>OS X 10.9</string>
    <key>MaximumDeploymentTarget</key>
    <string>10.9</string>
    <key>MinimalDisplayName</key>
    <string>10.9</string>
    <key>MinimumSupportedToolsVersion</key>
    <string>3.2</string>
    <key>SupportedBuildToolComponents</key>
    <array>
        <string>com.apple.compilers.gcc.headers.4_2</string>
    </array>
    <key>Version</key>
    <string>10.9</string>
    <key>isBaseSDK</key>
    <string>YES</string>
</dict>
</plist>
```

#####Example Sparse SDKSettings.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.
<plist version="1.0">
<dict>
    <key>AlternateSDK</key>
    <string>macosx10.9</string>
    <key>CanonicalName</key>
    <string>PrivateMacOSX10.9</string>
    <key>CustomProperties</key>
    <dict/>
    <key>DefaultProperties</key>
    <dict/>
    <key>DisplayName</key>
    <string>OS X Private (10.9)</string>
    <key>MaximumDeploymentTarget</key>
    <string>10.9</string>
    <key>MinimalDisplayName</key>
    <string>Private (10.9)</string>
    <key>MinimumSupportedToolsVersion</key>
    <string>3.2</string>
    <key>SupportedBuildToolComponents</key>
    <array>
        <string>com.apple.compilers.gcc.headers.4_2</string>
    </array>
    <key>Version</key>
    <string>Private (10.9)</string>
    <key>isBaseSDK</key>
    <string>NO</string>
</dict>
</plist>
```

↑ Table of Contents

SDK Contents

An SDK can contain anything necessary to perform a build. They typically contain frameworks, libraries, headers, resource, scripts/binaries, and any other type of asset that the build system would rely on.

####Frameworks, Libraries, and Headers Xcode will automatically search specific locations for any frameworks, libraries, and headers that have been included in the SDK bundle. These paths can differ based on the type of the SDK, additional paths can be specified by using the `-isystem` and `-iframework` flags to the compiler. Placing files in the locations listed below according to SDK type and if the files are a framework/library/headers to make

them be automatically found. (All paths are based on the root of the SDK bundle)

Type	Base SDK Search Paths	Sparse SDK
Frameworks	`/Library/Frameworks/`, `/System/Library/Frameworks/`	`/System/Li`
Libraries	`/usr/lib/`, `/usr/local/lib/`	`/usr/lib/`
Headers	`/usr/include/`, `/usr/local/include/`	`/usr/inclu`

####Assets Additional types of assets (graphics, fonts, bundles, strings files, etc) can be bundled as part of a SDK. There is no built-in support for finding any of these assets in the SDK. You will have to provide a way to access them from the SDK so they can be utilized.

####Executables While it is possible to ship scripts and executable binaries as part of an SDK bundle, Xcode's tooling doesn't know how to find these based on the specified SDK. To access these files you need to define the paths to run them directly since `xcrun` won't find them. If they are part of the Base SDK, then they can be found by using `$(SDKROOT)` as the base path or custom variables set via the SDKSettings.plist, however if they are part of a Sparse SDK then you will have to define the paths yourself.

↑ Table of Contents

Using with Xcode

#####Base SDK Xcode searches for Base SDKs based on the target platform bundles found in `/Platforms/` directory in the currently selected `DEVELOPER_DIR` path. The search path for SDKs in a platform bundle is `/Developer/SDKs/`, eg:

```
|-- /Applications/Xcode.app/Contents/Developer/Platforms/
    |-- MacOSX.platform
        |-- Developer
            |-- SDKs
                | -- MacOSX10.9.sdk
                | -- MacOSX10.10.sdk
    |-- iPhoneOS.platform
        |-- Developer
            |-- SDKs
                | -- iPhoneOS.sdk
    |-- iPhoneSimulator.platform
        |-- Developer
            |-- SDKs
                | -- iPhoneSimulator.sdk
```

The `xcodebuild` command line tool allows you to specify the SDK to use when building by providing a flag and either a path or name of an SDK (see CanonicalName in SDKSettings.plist). This tool will find SDKs specified by path, if they exist inside of the current `DEVELOPER_DIR` path. If you specify a SDK by path that exist outside of that directory then it will be ignore and fall back to the default SDK. To view the names of SDKs that are found by xcodebuild and Xcode you can run the following command:

```
$ xcodebuild -showsdks
OS X SDKs:
    OS X 10.9                       -sdk macosx10.9
    OS X 10.10                      -sdk macosx10.10

iOS SDKs:
    iOS 8.2                         -sdk iphoneos8.2

iOS Simulator SDKs:
    Simulator - iOS 8.2             -sdk iphonesimulator8.2
```

#####Sparse SDK Sparse SDKs must be specified using the `ADDITIONAL_SDKS` build setting. Multiple Sparse SDKs can be specified, they must be paths to the SDK bundle directory and each path must be separated by a space. The list is sorted by precedence, SDKs higher in the list will be searched first. If a header or library exists in more than one SDK, the first instance that is found will be used. The build system Xcode uses will resolve what additional flags must be added to compile.

Linking:

Linking libraries and frameworks from a Sparse SDK becomes a bit more opaque than linking a library that resides in a Base SDK. When adding a library or framework to link from a target's Build Phases panel, Xcode will only display a

list of libraries and frameworks that are in the Base SDK. You can use the "Add Other" button to navigate to the library you want to link, however this can lead to Xcode using absolute paths when adding the library as a reference to the project file. This will work fine until the Sparse SDK is moved or the path otherwise changes. To avoid this issue you can link the library by supplying the linker flags for it; either `-l <library name>` for stand alone libraries, or `-framework <framework name>` for frameworks. Since Xcode's build system will already be supplying the search paths for the Sparse SDK based on what includes you have, this will properly resolve and link.

Copying:

To handle copying resources or frameworks into an application bundle I would recommend either:

- setting up an xcconfig file that has relative paths to the Sparse SDK and has variables to the files that need to be copied so they can be accessed by a script to perform the copy into the application bundle before signing. Example xcconfig file:

```
// Additional SDKs.xcconfig

// defining variables for the first Sparse SDK that should be use
FOO_SDK = ~/Library/Application\ Support/Developer/Shared/Xcode/Sl
TESTING_FRAMEWORK = Testing
COPY_TESTINGFRAMEWORK_PATH = $(FOO_SDK)/System/Library/Frameworks/

// defining variables for the second Sparse SDK that should be us
BAR_SDK = ~/Library/Application\ Support/Developer/Shared/Xcode/Sl
UI_ADDITIONS_LIBRARY = UIAdditions
UI_ADDITIONS_LIBRARY_PATH = $(BAR_SDK)/usr/lib/lib$(UI_ADDITIONS_

// setting the ordering of the additional Sparse SDKs
ADDITIONAL_SDKS = $(FOO_SDK) $(BAR_SDK)

//  Configuring the linker flags to link the Testing framework fr
// to link the UIAdditions library from the second Sparse SDK
OTHER_LDFLAGS = $(inherited) -framework $(TESTING_FRAMEWORK) -l $

//   Now you have the variables $(COPY_TESTINGFRAMEWORK_PATH) and
// and you can add an additional script phase after the regular C
// these paths into the application bundle in the correct locatio
// after copying them manually like this depending on what your t
```

(Note: Please see my post about xcconfig files if you are unfamiliar with using xcconfig files)

- adding the Sparse SDK to the project file as a reference. This would allow you to link as part of the Link Libraries build phase, and not have to add linker flags for each library. Note that Xcode will add the reference as an absolute path by default, changing the path to be defined as a relative path to prevent it breaking can be difficult and may require editing the project.pbxproj file directly.

↑ Table of Contents

Related Radars

Radar	rdar://19969415
Status	Open
Title	xcrun does not use SDK specific search paths
Details	xcrun will not search active or specified SDKs for tools, it only searches the platform bundle's paths.

Radar	rdar://21426816
Status	Open
Title	Sparse SDK documentation
Details	There is no document that outlines how Sparse SDKs work with Xcode. How Sparse SDKs are used has changed since their introduction and this remains an undocumented yet supported feature in the shipping version of Xcode.

Radar	rdar://21426869
Status	Open
Title	mksdk tool is not updated
Details	The command line tool `mksdk` hasn't been properly updated to work with how Sparse SDKs are used now. This tool needs to be updated to help create SDKs.

Radar	rdar://21426912
Status	Open
Title	mksdk documentation
Details	The usage and help for the `mksdk` tool is very vague and doesn't give a clear idea as to how to use the tool properly. A manual page and updated usage would be helpful.

Radar	rdar://21426990
Status	Open
Title	Define where Base SDKs can exist
Details	The location of where Base SDKs (System SDKs) can exist and be found by the Xcode build system isn't defined in any of the developer tools documentation. Defining how SDK bundles are found and processed would be very useful.

Radar	rdar://21427092
Status	Open
Title	Define what search paths are used for contents of Sparse SDKs vs Base SDKs
Details	The search paths for contents of a Sparse SDK vs a Base SDK differ. These differences and why they exist are not documented.

Radar	rdar://21427142
Status	Open
Title	Sparse SDKs Framework search paths
Details	Frameworks are not search for in `/Library/Frameworks/` in a Sparse SDK.

Radar	rdar://21427166
Status	Open
Title	Sparse SDK Header Search Paths
Details	Headers are not searched for in `/usr/local/include/` in Sparse SDKs.

Radar	rdar://21427221
Status	Open
Title	xcodebuild -sdk flag doesn't accept some SDKs
Details	xcodebuild's usage describes the `-sdk` flag as being able to specify the CanonicalName of an SDK or the path to an SDK. However the tool will only accept paths to SDKs that exist inside of the currently selected developer directory (found by running `xcode-select -p`.

Radar	rdar://21427291
Status	Open
Title	Xcode doesn't list libraries from Sparse SDKs
Details	When adding a linked library to a target's linking build phase, the list of libraries shown in the drop down doesn't include libraries that are found inside of Sparse SDKs. it will list libraries from the Base SDK and the developer directory, but not any supplemental SDKs that are used with the target.

Radar	rdar://21427330
Status	Open
Title	Support for file references in Xcode using relative paths
Details	There should be built-in support for Xcode add files by relative path of the user's home directory `~/` rather than creating absolute paths when adding new files.

Radar	rdar://21427411
Status	Open
Title	DefaultProperties and CustomProperties importing from Sparse SDK SDKSettings.plist files
Details	Currently Sparse SDKs do not support creation or modification of variables based on the DefaultProperties and CustomProperties keys in the SDKSettings.plist. Base SDKs do support this behavior. Sparse SDKs should be able to override and set default settings as well.

↑ Table of Contents