Xcode Build Locations

This is a guide on how to resolve the on-disk location of the location on the file-system that Xcode will use when performing a **build** operation. To understand how to use this information, you should be familar with the Build Locations section of my guide "Managing Xcode".

The build location preference that you configure in Xcode is stored in the `com.apple.dt.Xcode` preferences file. To read and write values to this file you should be using the command line tool `defaults`, that comes with OS X.

Getting the Derived Data Location

The first value that must be fetched is the location of the Derived Data directory. There is a default path for this:

```
$(HOME)/Library/Developer/Xcode/DerivedData/
```

However, the user can configure Xcode to use an alternative location for this path as well. To find if there is a user configured path, perform the following query:

```
$ defaults read com.apple.dt.Xcode IDECustomDerivedDataLocation
```

If there is no value set for this key, then the path for the Derived Data location should follow the default pattern mentioned above. If this value is set, then it is interpreted as a relative path to the Xcode project file directory.

```
$(PROJECT_DIR)/$(IDECustomDerivedDataLocation)
```

Note: For the rest of this post the resulting value (default or custom) that should be used for the path to the Derived Data location will be referred to as `$(DERIVED_DATA_PATH)`.

Getting the Type of Build Location

The next value that needs to be queried for is the type of the build location. This can be done by performing:

```
$ defaults read com.apple.dt.Xcode IDEBuildLocationStyle
```

This will return one of the following options:

- `Unique`
- `Shared`
- `Custom`
- `DeterminedByTargets`

Each location type follows different patterns for composing the path used for builds.

Unique

Xcode uses the **unique** location option by default, and this is composed as the following:

```
$(DERIVED_DATA_PATH)/$(PROJECT_NAME)-$(PROJECT_PATH_HASH)/Build/Product
```

Where:

- `$(PROJECT_NAME)` is the name of the Xcode project or workspace file (without extension)
- `$(PROJECT_PATH_HASH)` is the hash that is described in this blog post

Shared

The **shared** build location is an option to have all built products of all Xcode projects placed in a shared location. To find this location, perform the following query:

```
$ defaults read com.apple.dt.Xcode IDESharedBuildFolderName
```

This will return a relative path that should be composed with the Derived Data path to get the location of the build directory on the file-system.

```
$(DERIVED_DATA_PATH)/$(IDESharedBuildFolderName)
```

Custom

The **custom** build location option has a lot of ways that it can be configured. If this is set then two additional queries must be performed:

```
$ defaults read com.apple.dt.Xcode IDECustomBuildLocationType
```

To determine how the custom path should be composed.

```
$ defaults read com.apple.dt.Xcode IDECustomBuildProductsPath
```

To determine what the custom path is.

The `IDECustomBuildLocationType` key can be set to:

- `RelativeToDerivedData` - In this case the custom path is a relative path based on the location of the Derived Data directory:

```
$(DERIVED_DATA_PATH)/$(IDECustomBuildProductsPath)
```

- `RelativeToWorkspace` - In this case the custom path is a relative path based on the location of the workspace that is being built:

```
$(WORKSPACE_OR_PROJECT_DIR)/$(IDECustomBuildProductsPath)
```

- `Absolute` - In this case the custom path is an absolute path on the file-system:

```
$(IDECustomBuildProductsPath)
```

DeterminedByTargets

This is called the **Legacy** option in Xcode's preferences. What this does is falls back to the old default behavior that Xcode had for building targets in a relative path set by the `SYMROOT` value in the Xcode project itself. The path gets composed as such:

```
$(PROJECT_DIR)/$(SYMROOT)
```