How to solve symbolication problems

General

In most cases the symbolication process doesn't work, since there is no dSYM uploaded to HockeyApp, or the dSYM doesn't match the application binary. So this is also about the rules of binary UUIDs and dSYMs.

IMPORTANT: Each time you run the build command, your app gets a new unique UUID which is placed into the crash report to idenfiy the build. You also get a new dSYM package which contains the same UUID. So if you upload a new binary to the app store, you also have to upload the new dSYM to HockeyApp!

About UUIDs

Here are some tips on how to find out which UUID is used where:

1. Find the UUID in the crash report:

   - Scroll down the crash report until you find `Binary Images:`.
   - The first line below that shows something like the following:

     ```
     0x1000 -   0x222fff +AppName armv7  <1234567890abcdef1234567890abcdef> /var/mobile/Applicati
     ```

   `1234567890abcdef1234567890abcdef` is the UUID of your binary for the `armv7` architecture.

2. Find the UUID in the app binary (1 line for each architecture):

   ```
   dwarfdump --uuid AppName.app/AppName
   ```

   The result will look like:

   ```
   UUID: 12345678-90AB-CDEF-1234-567890ABCDEF (armv7) AppName.app/AppName
   ```

3. Find the UUID in the dSYM (1 line for each architecture):

   ```
   dwarfdump --uuid AppName.app.dSYM
   ```

   The result will look like:

   ```
   UUID: 12345678-90AB-CDEF-1234-567890ABCDEF (armv7) AppName.app.dSYM/Contents/Resources/DWARF/AppN
   ```

4. Find the dSYM for a specific UUID on your computer:

   ```
   mdfind "com_apple_xcode_dsym_uuids == 12345678-90AB-CDEF-1234-567890ABCDEF"
   ```

   The string "12345678-90AB-CDEF-1234-567890ABCDEF" is the UUID string from the crash report reformatted to uppercase and 8-4-4-4-12 groups.

If you found the correct dSYM, please upload it again and HockeyApp will process the crash logs a second time.

Build settings for getting proper symbol data

To get filenames, class names and line numbers in the crash reports, you need to make sure the build settings are setup properly depending on the build target type.

1. For static libraries the build settings contain the following:

   ```
   Strip Debug Symbols During Copy: No
   Strip Style: Debugging Symbols
   Strip Linked Product: No
   ```

2. For application targets or OS X frameworks, the build settings should contain the following for the release (App Store) build configuration:

   ```
   Strip Debug Symbols During Copy: Yes
   Strip Style: All Symbols
   ```

```
Strip Linked Product: Yes
```

Bitcode

When uploading an app to the App Store and leaving the "Bitcode" checkbox enabled, Apple will use that Bitcode build and re-compile it on their end before distributing it to devices. This will result in the binary getting a new UUID and there is an option to download a corresponding dSYM through Xcode.

Please note: Bitcode is not necessary to profit from other App Thinning features such as slicing! Both features work independently from each other.

To download the dSYMs for Bitcode enabled apps please open the Xcode's Organizer. Select the specific Archive of your app that you uploaded to iTunes Connect and click on the "Download dSYMs" button which will insert the Bitcode compiled dSYMs into the original archive. Then manually upload the symbols to the corresponding app and version on HockeyApp.

If the Xcode organizer doesn't provide any new symbols, you need to download the dSYM from the iTunes Connect portal. Please follow these steps:

- Select your app
- Select the Activity tab on top
- Select the build version of your app which has the missing symbols
- Click the Download dSYM link
- Upload the file you downloaded to HockeyApp (It is a zip file with the symbols included)

If you want to know how to manually upload symbols follow our How to manually upload and symbolicate a crash guide.