3rd Party Native Libraries

Steve Hannah
Codename One
Sept. 24, 2015

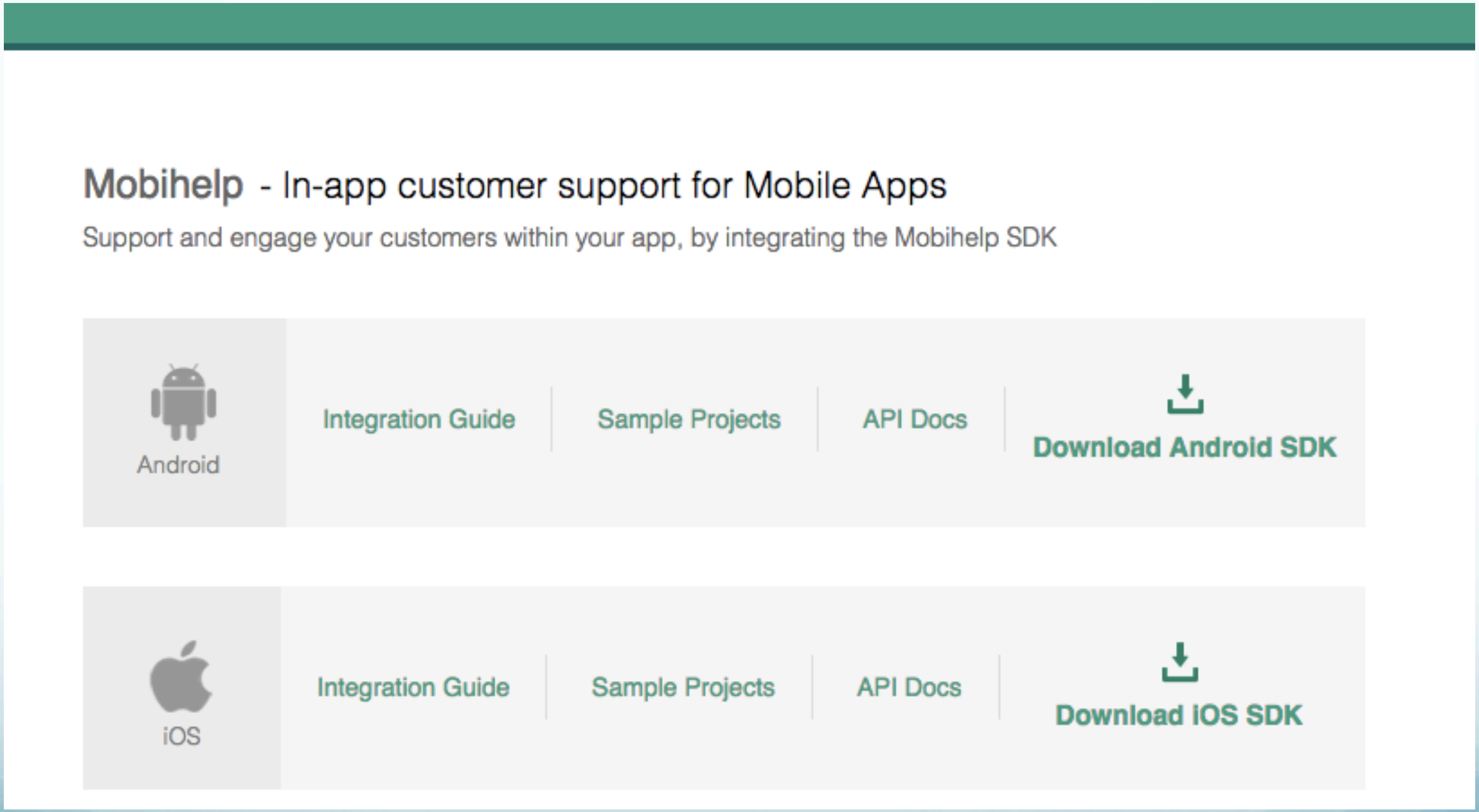
Making the FreshDesk Library

1. Review FreshDesk SDKs
2. Design Public API
3. Implement Public API and Native Interfaces
4. For Each Platform P in {Android, iOS}
   1. Implement Native interfaces.
   2. Bundle Native SDK and Dependencies
   3. Add native build hints
5. Package as CN1Lib

Reviewing FreshDesk SDKs

Want to answer two questions:

1. What should my public API look like?
2. What will be involved in integrating native SDK in my app or lib?

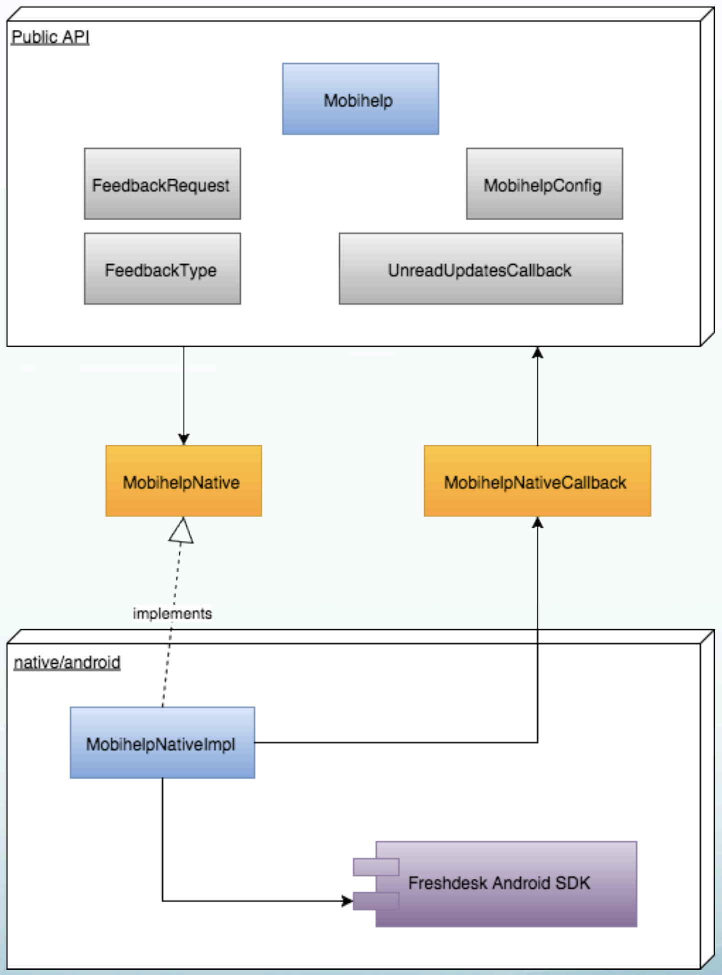


Mobihelp - In-app customer support for Mobile Apps

Support and engage your customers within your app, by integrating the Mobihelp SDK

http://developer.freshdesk.com/mobihelp/

The Architecture



Implementing Public API

We will mirror the Android API very closely

http://developer.freshdesk.com/mobihelp/android/api/reference/com/freshdesk/mobihelp/package-summary.html

Things to consider:

1. Native Interfaces only accept primitive parameters and return types (& Strings)
2. Some parameters in Android API are android-specific (e.g. Context, Activity)
3. How to handle callbacks

Non-primitive Parameters

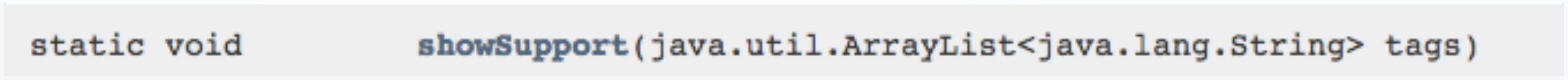
- Public API may have non-primitive parameters, but…
- Ultimately parameter will have to be piped through native interface in some form – so keep that in mind.

Strategies for Non-Primitives

1. Encode parameter as String or byte[] that can be parsed on native side.
2. Convert parameter to ID or token that can be passed and looked up later.
3. Separate compound types into primitive components, and pass individually.

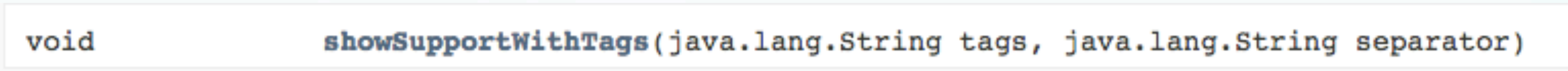
Example: Encoding ArrayList as String

- Public API:



```
static void          showSupport(java.util.ArrayList<java.lang.String> tags)
```

- - > Wrapped Native Interface Method:



```
void            showSupportWithTags(java.lang.String tags, java.lang.String separator)
```

Example: Android Specific Params

- Many methods in Android API take android.content.Context parameter
  - This is Android specific, so we can’t include in public API.
  - Inject in native layer and omit from our public API
    - Codename One provides class AndroidNativeUtil that provides access to the application Activity and Context.

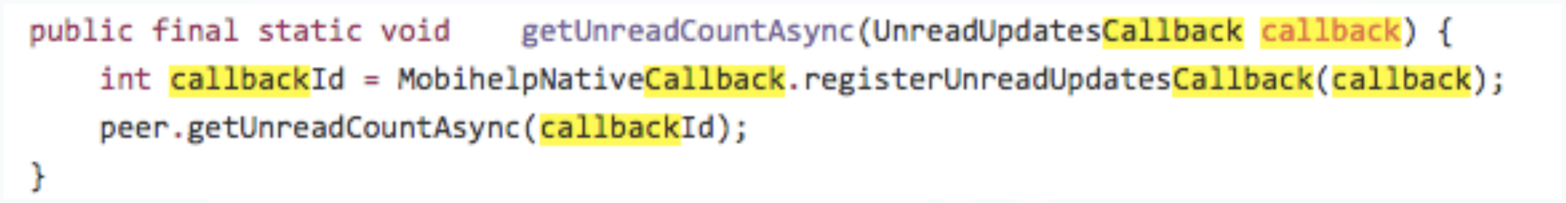
```java
public void showSolutions() {
    activity().runOnUiThread(new Runnable() {
        public void run() {
            com.freshdesk.mobihelp.Mobihelp.showSolutions(context());
        }
    });

}
```

```java
private static Context context() {
    return com.codename1.impl.android.AndroidNativeUtil.getActivity().getApplicationContext();
}
```

Example: Using Tokens

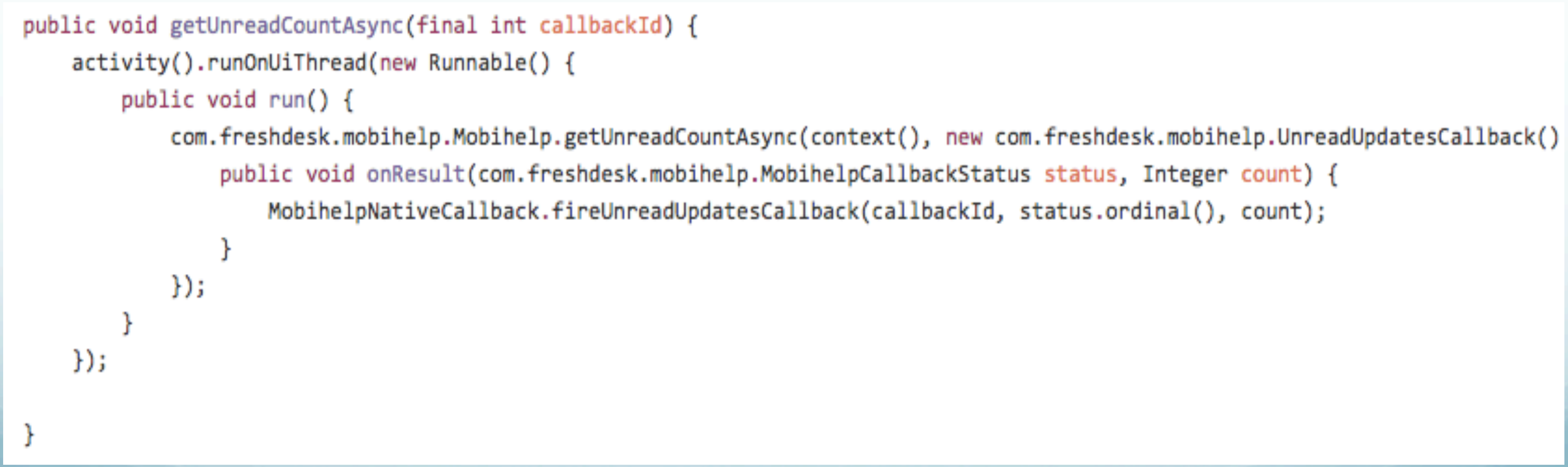
- Problem: Can’t pass a callback like a Runnable through Native Interface
- Solution: Store Runnable in static lookup table, and pass the ID or token.
  - Additionally provide static utility methods to make it easier to call from the native layer.

Public API Implementation:



```
public final static void    getUnreadCountAsync(UnreadUpdatesCallback callback) {
    int callbackId = MobihelpNativeCallback.registerUnreadUpdatesCallback(callback);
    peer.getUnreadCountAsync(callbackId);
}
```

Native Interface Implementation:



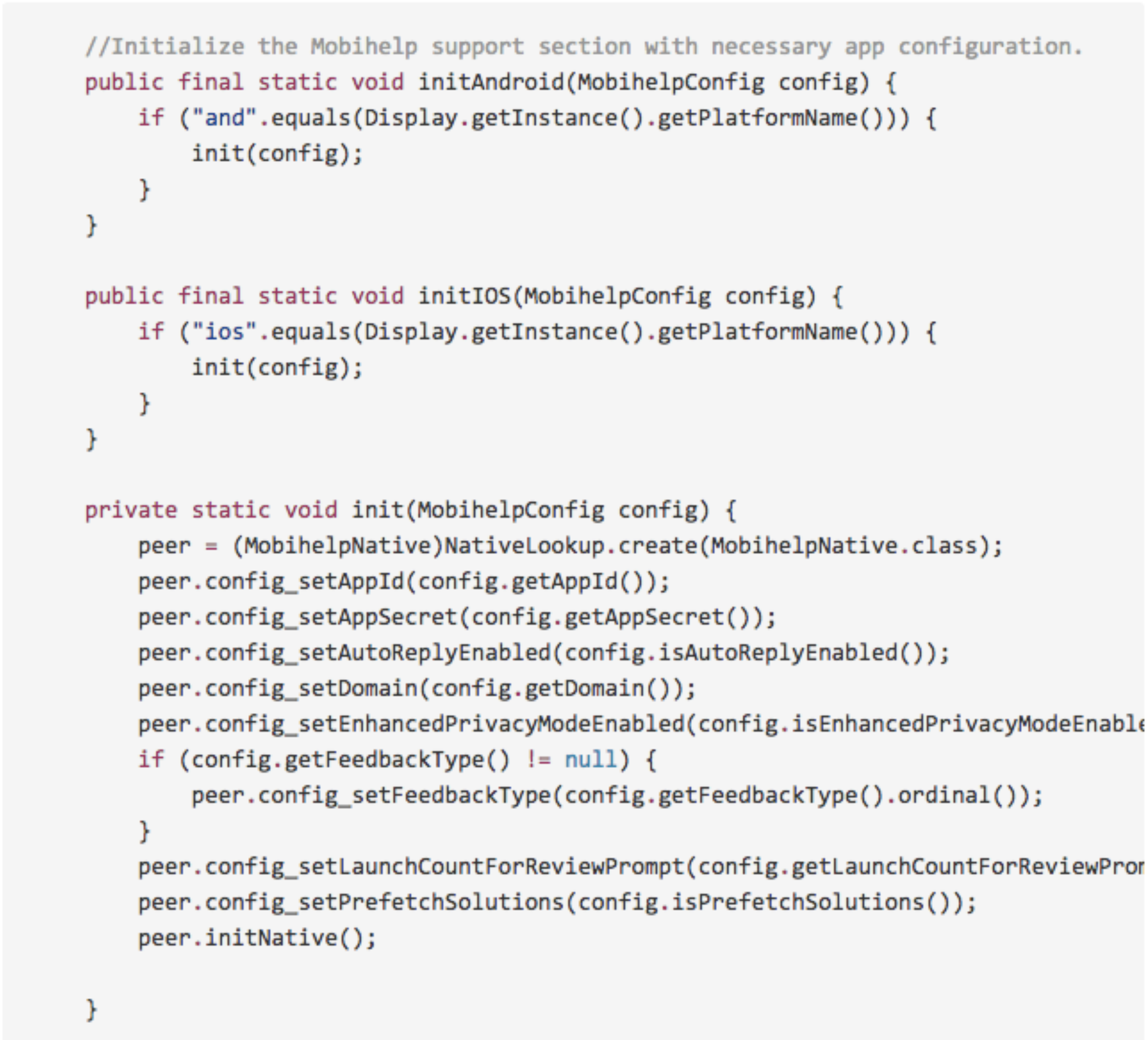
```
public void getUnreadCountAsync(final int callbackId) {
    activity().runOnUiThread(new Runnable() {
        public void run() {
            com.freshdesk.mobihelp.Mobihelp.getUnreadCountAsync(context(), new com.freshdesk.mobihelp.UnreadUpdatesCallback()
                public void onResult(com.freshdesk.mobihelp.MobihelpCallbackStatus status, Integer count) {
                    MobihelpNativeCallback.fireUnreadUpdatesCallback(callbackId, status.ordinal(), count);
                }
            });
        }
    });

}
```

Initialization

- Problem: FreshDesk gives you different API keys for each platform – but their API only has place to specify one key (for current platform)
- Solution: Create separate initIOS() and initAndroid() methods.

Implementing Glue Public API -> Native Interface

- Store reference to native API instance inside Public API class.
  - Methods are thin wrappers around native object
- Use NativeLookup class to initialize native peer.



```java
//Initialize the Mobihelp support section with necessary app configuration.
public final static void initAndroid(MobihelpConfig config) {
    if ("and".equals(Display.getInstance().getPlatformName())) {
        init(config);
    }
}

public final static void initIOS(MobihelpConfig config) {
    if ("ios".equals(Display.getInstance().getPlatformName())) {
        init(config);
    }
}

private static void init(MobihelpConfig config) {
    peer = (MobihelpNative)NativeLookup.create(MobihelpNative.class);
    peer.config_setAppId(config.getAppId());
    peer.config_setAppSecret(config.getAppSecret());
    peer.config_setAutoReplyEnabled(config.isAutoReplyEnabled());
    peer.config_setDomain(config.getDomain());
    peer.config_setEnhancedPrivacyModeEnabled(config.isEnhancedPrivacyModeEnable
    if (config.getFeedbackType() != null) {
        peer.config_setFeedbackType(config.getFeedbackType().ordinal());
    }
    peer.config_setLaunchCountForReviewPrompt(config.getLaunchCountForReviewPron
    peer.config_setPrefetchSolutions(config.isPrefetchSolutions());
    peer.initNative();

}
```

Hands-On

- Demo creating project (so far) in NetBeans
- Up to & Including Android Native Implementation

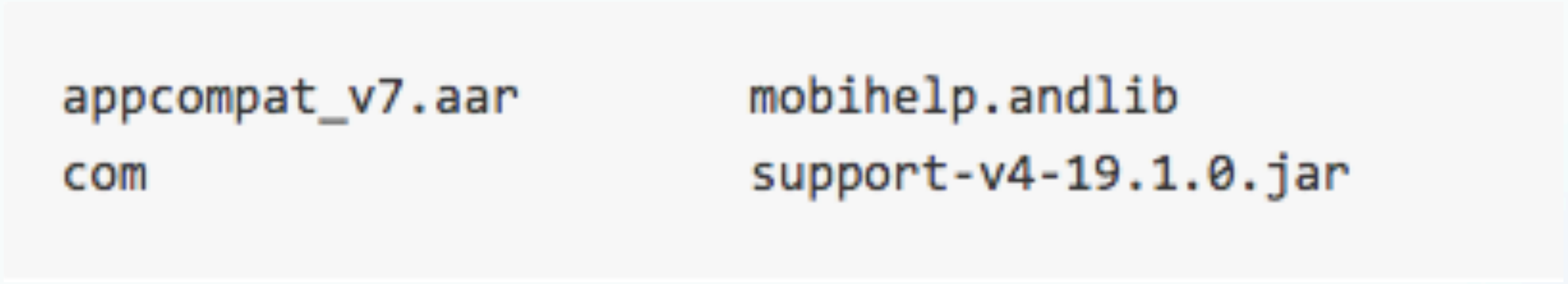
Bundling the Native SDKs

- Simplified procedure:
  - Copy native 3rd party dependencies into the native/android directory
- Long version:
  - Copy .jar deps into native/android
  - Libraries that include other resources like GUI XML files, etc… can be distributed as zipped eclipse/android studio projects with extension changed to ".andlib" … or as .aar files.

FreshDesk Android SDK

- Includes resources... distributed as Eclipse/ Android Studio project.
- Depends on AppCompat_v7 library
  - Need to download Android Support Library version 19, retrieve .aar from local maven repo, and copy to native/android directory.
  - AppCompat_v7 depends on appsupport-v4.

android/native directory



```
appcompat_v7.aar          mobihelp.andlib
com                       support-v4-19.1.0.jar
```

Adding Build Hints

- Inject necessary items into AndroidManifest.xml
  - Use android.xapplication hint for adding items to <application> tag.
  - Use android.xpermissios to add <uses-permission> tags.
- Set up Proguard keep rules:
  - Use android.proguardKeep build hint

Hands On

- Add android native SDKs and build hints

Troubleshooting

- Select “include source” option in build hints to download Eclipse project that you can debug in eclipse.

Package as .cn1lib

1. I usually do initial development in regular project for convenience.
2. Better to package as cn1lib when done.
3. Process
   1. Create new Codename One Library Project
   2. Copy .java files from original to library project
   3. Copy native directory from original to library project.
   4. Copy *relevant* build hints from codenameone_settings.properties into lib project’s codenameone_library _appended.properties

Hands-On

- Create cn1lib project.

Next Time

- We'll look at the iOS side of the native interface.