MobileFirst Quality Assurance for native Android applications

Overview

Prerequisite: This tutorial is a continuation of the MobileFirst Quality Assurance (../overview/) tutorial. To follow it, you must first set up IBM Mobile Quality Assurance (MQA).

This tutorial explains how to install and configure the Mobile Quality Assurance client SDK for Android, and how to enable apps to send back bug reports and feedback.

Jump to:

Creating a simple Android app

If you do not already have an Android app to use for this procedure, create a Hello World app by completing the following steps in Android Studio:

1. Open Android Studio and select **File** > **New** > **New project**.
2. Name your new application *Helloworld*.
3. Identify the location where you want your project files stored, or select the default path. Click **Next**.
4. Select **Phone and Tablet** for your device type and accept the default minimum SDK of Android that is listed. Click **Next**.
5. Select the blank activity type, and click **Next**.
6. You can leave the default values in the Customize the Activity window for this example. Click **Finish**.
7. After the app builds, you can test it by running it in an emulator by selecting **Run 'app'** from the **Run** menu.

Installing the Android libraries

The libraries support both preproduction mode or production mode. Use preproduction mode when you want internal testers to report bugs and feedback with detailed information and have apps under test automatically report crashes. Complete the procedure for production mode when the app is publicly available and you want to gather information after it is released. Complete these steps when using the Android platform:

1. Download the most recent version of the Mobile Quality Assurance SDK for Android. For more information about downloading the SDK, see Android SDK for download (http://www.ibm.com/support/knowledgecenter/SSJML5_6.0.0/com.ibm.mqa.uau.saas.doc/topics/c_AndroidSDKsForDownload.html). **Important:** Ensure that you download the SDK that corresponds to your Android IDE, either Android Studio or Android Development Tools (ADT).
2. Add Mobile Quality Assurance to your project by using Android Studio.
    1. Click **File** > **New Module**.
    2. In the Create New Module wizard, under *More Modules*, select **Import .JAR or .AAR Package**, then click **Next**.
    3. In the File name field, enter the name of the Mobile Quality Assurance SDK `.aar` file that you downloaded. **Tip:** To browse to the file, click **...**.
    4. Click **Finish**.
3. Add Mobile Quality Assurance to your mobile app by using Android Studio:
    1. Click **File** > **Project Structure**.
    2. In the Project Structure window, select the module for your app, and then select the Dependencies tab.
    3. Click **+** > **Module Dependency**.
    4. In the Choose Modules window, select the Mobile Quality Assurance Android module that you imported, and then click **OK**. The Choose Modules window closes.
    5. Click **OK** to close the Project Structure window.
4. Update the `AndroidManifest.xml` file: **Tip:** The manifest file included with the SDK contains all the activities and permissions for Mobile Quality Assurance. The optional permissions are commented out. To enable them, remove the < -- and --> tags that surround each permission.
    1. Declare the Mobile Quality Assurance preproduction or production activities. Ensure that the manifest file includes an application that contains the following activities:

```
            <activity android:name="com.ibm.mqa.ui.ProblemActivity"></activity>
            <activity android:name="com.ibm.mqa.ui.FeedbackActivity"></activity>
            <activity android:name="com.ibm.mqa.ui.ScreenshotEditorActivity"></activity
>
```

    2. Ensure that the following required permissions are listed in the manifest file:
        - INTERNET - Mobile Quality Assurance uses this permission to connect with the IBM® Mobile Quality Assurance for Bluemix server. The following code snippet shows the INTERNET permission entry that you must include:

```
<uses-permission android:name="android.permission.INTERNET" />
```

- SYSTEM_ALERT_WINDOW - **Note:** This permission is required as of Android 5.1. Mobile Quality Assurance requires this permission to open windows on top of your application, such as the Mobile Quality Assurance app tester log-in window in preproduction mode, the Mobile Quality Assurance bug and feedback reporting window in preproduction mode, and the Mobile Quality Assurance feedback window in production mode. The following code snippet shows the `SYSTEM_ALERT_WINDOW` permission entry that you must include:

```
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
```

Important: If you do not add the permission, the app might fail with an `InvalidDisplayException` error: In preproduction mode, the app might fail to start. In production mode, if the app uses the `MQA.feedback()` API, it might crash when the API is called.

- For optional settings that you can enable, see Installing the Android libraries (http://www-01.ibm.com/support/knowledgecenter/SSJML5_6.0.0/com.ibm.mqa.uau.saas.doc/topics/t_DownloadingAndInstallingTheAndroidLibraries.html) in IBM Knowledge Center.

Configuring how MQA communicates with your app

1. Configure MQA to start a new Mobile Quality Assurance session each time your app starts in the Java file:
   1. Ensure that Mobile Quality Assurance is imported by verifying that your code includes the following lines:

```
import com.ibm.mqa.MQA;
import com.ibm.mqa.MQA.Mode;
import com.ibm.mqa.config.Configuratio
n;
```

   2. Build a configuration object that specifies how Mobile Quality Assurance behaves.
   3. Call the `MQA.startNewSession()` method in the `onCreate` event of your main activity.

```
public class MainActivity extends Activity {
  public static final String APP_KEY = "Your-Application-Key-Goes-Here";

  @Override public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Configuration configuration = new Configuration.Builder(this)
   .withAPIKey(APP_KEY); // Provides the quality assurance application APP_KE
Y
    .withMode(Mode.QA)  // Selects the quality assurance application mode
    .withReportOnShakeEnabled(true) // Enables shake report trigger
    .build();
    MQA.startNewSession(MainActivity.this, configuration);
  }
}
```

> To learn more about starting a Mobile Quality Assurance session, including details about the different session configuration options, see Starting an Android session (http://www.ibm.com/support/knowledgecenter/SSJML5_6.0.0/com.ibm.mqa.uau.saas.doc/topics/t_StartingAnAndroidSession.html).

   4. Add your valid application key to send data to Mobile Quality Assurance. **Tip:** If you already have an account, you can find your app key in the Mobile Quality Assurance web pane under App Settings. In the code snippet in which you created the configuration object, replace the *Your-Application-Key-Goes-Here* variable with your application key from Mobile Quality Assurance.
2. Send your log calls to Mobile Quality Assurance. Mobile Quality Assurance comes with its own logging library *com.ibm.mqa.Log*, which you can use in place of the standard Android logging library. To have all of your logs go through Mobile Quality Assurance, replace the `import android.util.Log;` import statement with: `import com.ibm.mqa.Log;`.
3. Gather user feedback (preproduction only). If your app is in preproduction with Mobile Quality Assurance, you can get user feedback from within the app.

Testing your app

After the environment is set up, you can load your app on your Android device and start testing it. Complete the following steps to report a bug or provide feedback during testing:

1. Shake your mobile device lightly if the shake option is enabled. If it is not enabled, you can tap the MQA icon on your notifications bar.
2. Select **Report a bug** or **Give feedback**.
3. Enter the required information and select **Send** to send the information.
4. View the bug reports and feedback in your Bluemix MQA instance where you retrieved the AppKey.

For more information about reporting bugs, see Reporting bugs on Android devices

(https://www.ibm.com/support/knowledgecenter/#!/SSFRDS_6.3.0/com.ibm.mqa.uau.doc/topics/t_ReportingBugsOnAndroidDevices.html). For more information about submitting feedback, see Submitting feedback on an Android device (https://www.ibm.com/support/knowledgecenter/#!/SSFRDS_6.3.0/com.ibm.mqa.uau.doc/topics/tandroidFeedback.html).

Further discovery

After you are comfortable with the basics of setting up MQA with your app, you can read content in IBM Knowledge Center to learn and explore some of the other things that you can do. For example:

- Managing users and testers (http://www.ibm.com/support/knowledgecenter/SSJML5_6.0.0/com.ibm.mqa.uau.saas.doc/topics/t_ManagingTesters.html). You can manage the roles and access of your team as they apply to your app and MQA instance.
- Distributing and managing builds (http://www.ibm.com/support/knowledgecenter/SSJML5_6.0.0/com.ibm.mqa.uau.saas.doc/topics/t_DistributingBuilds.html). You can distribute test builds and production builds to the appropriate audience.
- Managing reports (http://www.ibm.com/support/knowledgecenter/SSJML5_6.0.0/com.ibm.mqa.uau.saas.doc/topics/t_ManagingReports.html). You can automatically distribute test builds and production builds to your audience.
- User sentiment analytics (http://www.ibm.com/support/knowledgecenter/SSJML5_6.0.0/com.ibm.mqa.uau.saas.doc/topics/UserSentiment.html). You can use MQA to analyze and monitor user comments and information about your app. You can also compare the analysis of those comments with the analysis of the user feedback from similar apps.