AT&T Developer Program

Application Resource Optimization

Using the ARO Data Collector in an Android Application

Revision Date **December 28, 2012**

Legal Disclaimer

This document and the information contained herein (collectively, the "**Information**") is provided to you (both the individual receiving this document and any legal entity on behalf of which such individual is acting) ("**You**" and "**Your**") by AT&T, on behalf of itself and its affiliates ("**AT&T**") for informational purposes only. AT&T is providing the Information to You because AT&T believes the Information may be useful to You. The Information is provided to You solely on the basis that You will be responsible for making Your own assessments of the Information and are advised to verify all representations, statements and information before using or relying upon any of the Information. Although AT&T has exercised reasonable care in providing the Information to You, AT&T does not warrant the accuracy of the Information and is not responsible for any damages arising from Your use of or reliance upon the Information. You further understand and agree that AT&T in no way represents, and You in no way rely on a belief, that AT&T is providing the Information in accordance with any standard or service (routine, customary or otherwise) related to the consulting, services, hardware or software industries.

AT&T DOES NOT WARRANT THAT THE INFORMATION IS ERROR-FREE. AT&T IS PROVIDING THE INFORMATION TO YOU "AS IS" AND "WITH ALL FAULTS." AT&T DOES NOT WARRANT, BY VIRTUE OF THIS DOCUMENT, OR BY ANY COURSE OF PERFORMANCE, COURSE OF DEALING, USAGE OF TRADE OR ANY COLLATERAL DOCUMENT HEREUNDER OR OTHERWISE, AND HEREBY EXPRESSLY DISCLAIMS, ANY REPRESENTATION OR WARRANTY OF ANY KIND WITH RESPECT TO THE INFORMATION, INCLUDING, WITHOUT LIMITATION, ANY REPRESENTATION OR WARRANTY OF DESIGN, PERFORMANCE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, OR ANY REPRESENTATION OR WARRANTY THAT THE INFORMATION IS APPLICABLE TO OR INTEROPERABLE WITH ANY SYSTEM, DATA, HARDWARE OR SOFTWARE OF ANY KIND. AT&T DISCLAIMS AND IN NO EVENT SHALL BE LIABLE FOR ANY LOSSES OR DAMAGES OF ANY KIND, WHETHER DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, PUNITIVE, SPECIAL OR EXEMPLARY, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, LOSS OF GOODWILL, COVER, TORTIOUS CONDUCT OR OTHER PECUNIARY LOSS, ARISING OUT OF OR IN ANY WAY RELATED TO THE PROVISION, NON-PROVISION, USE OR NON-USE OF THE INFORMATION, EVEN IF AT&T HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSSES OR DAMAGES.

Table of Contents

1. Introduction

The Application Resource Optimizer (ARO) is a free diagnostic tool for analyzing mobile web application performance that was developed by AT&T. ARO automatically profiles an application, and provides recommendations to help optimize its speed and performance, make battery utilization more efficient, and reduce network impact.

The ARO Data Collector is the component of ARO that captures the data traffic of a mobile device and stores that information in trace files that can be analyzed using the ARO Data Analyzer. The ARO Data Analyzer evaluates the trace data collected from an application and generates statistical and analytical results based on recommended best practices.

The open source code for ARO is provided in a github repository at https://github.com/attdevsupport/ARO. This allows you to integrate the components of ARO, like the ARO Data Collector, into your own testing and development processes.

This document describes how to integrate ARO into an Android application by setting up an Android library project that exposes the ARO Data Collector.

An Android library project is a development project that holds shared Android source code and resources. Other Android application projects can reference the library project and, at build time, include its compiled sources in their .apk files.

The following sections of this document describe how to set up an ARO Data Collector Android Library project using the Eclipse IDE, and describes how to use that library in an Android application.

2. ARODataCollector Library Methods

The ARO Data Collector Android Library - *ARODataCollectorLib* – that is described in this document, exposes the following methods in the **ARODataCollector** class:

1. **startARODataCollector** – This method starts the ARO Data Collector. It takes the following parameters:

Parameter	Description
Context ***mAppContext***	An application project context.
PackageManager ***mAppPackageManager***	The package manager of the application project context.
String *mTraceFolderName*	The name of the folder in which to store the collected trace files. The contents of this folder will be overwritten if they are not empty, and special characters will not be allowed in the folder name.
Boolean *mVideoCapture*	Indicates whether a video of the trace should be captured.

2. **stopARODataCollector**—This method stops the ARO Data Collector. It takes no parameters.

3. Setting up the ARODataCollectorLib Android Library Project

The following steps describe how to include *ARODataCollectorLib* in your existing Android app, using the Eclipse IDE.

1. Check out the source code for the *ARODataCollectorLib* into the same workspace as your existing Android app.

2. On your existing Android app, right click “Properties” to open the “Properties” dialog, and click on “Android” in the left pane (as shown in the following figure).

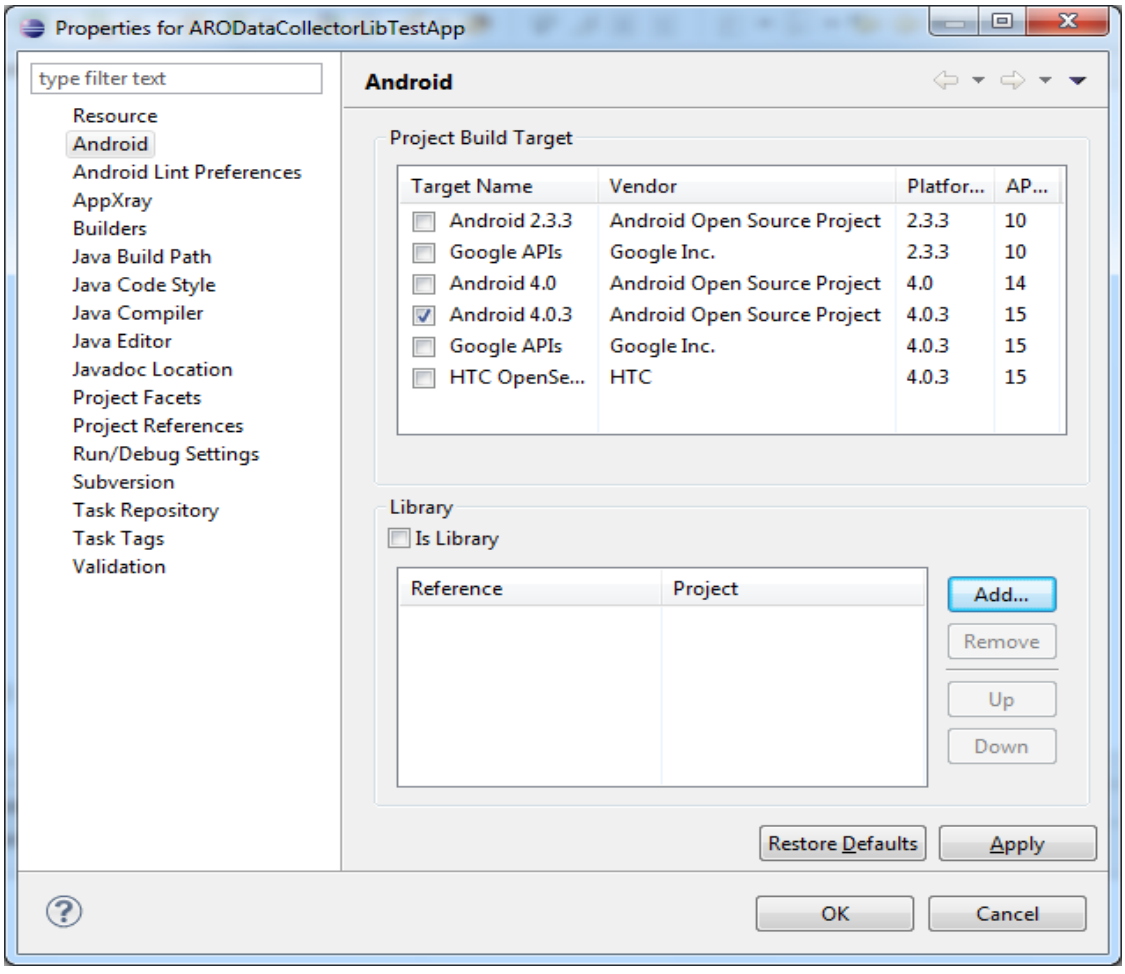

Figure 1 – The Properties dialog for ARODataCollectorLibTestApp.

3. In the “Library” pane on the right bottom half of the “Properties” dialog, click the “Add” button and select *ARODataCollectorLib* to include the library in your application.

Update the manifest file for your existing Android app to use *ARODataCollectorLib* by including the following permissions:

```
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_SETTINGS" />
    <uses-permission android:name="android.permission.ACCESS_SUPERUSER" />
    <uses-permission android:name="android.permission.BATTERY_STATS" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.GET_TASKS" />
    <uses-permission android:name="android.permission.RESTART_PACKAGES" />
    <uses-permission android:name="android.permission.KILL_BACKGROUND_PROCESSES"
/>
```

4. Using ARODataCollectorLib in an Application

This section describes an example application called ARODataCollectorLibTestApp that demonstrates how a dependent application can use the *ARODataCollectorLib* described in this document.

The following figure shows a completed set up of the ARODataCollectorLibTestApp and the *ARODataCollectorLib* in the Eclipse IDE.

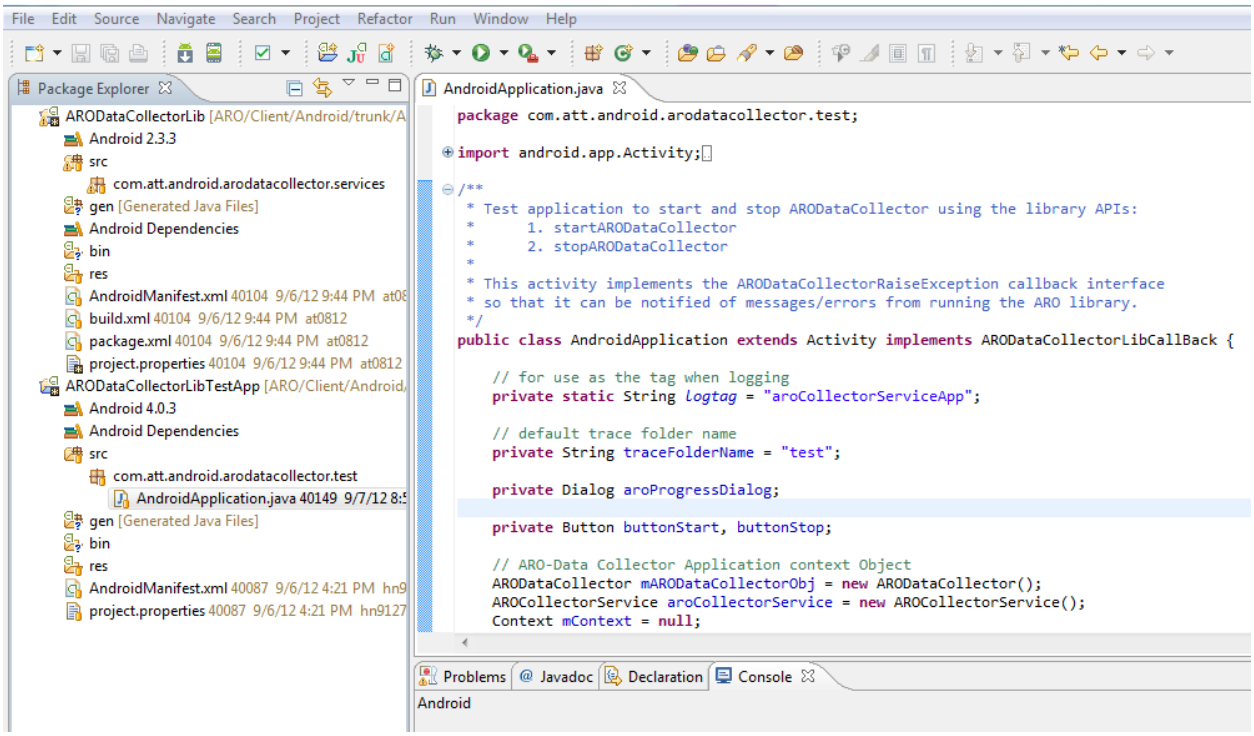

***Figure 2 – The ARODataCollectorLibTestApp* in Eclipse.**

The ARODataCollectorLibTestApp creates a simple UI that provides access to the exposed start and stop methods in *ARODataCollectorLib*, and implements callback methods to handle status notifications and exceptions from *ARODataCollectorLib.*

The ARODataCollectorLibTestApp implements a main class called **AndroidApplication.java** that does the following:

1. Extends the base class **android.app.Activity**.
2. Implements an *ARODataCollectorLibCallBack* interface to receive status notifications exceptions from the *ARODataCollectorLib* library.
3. Sets up the Start and Stop buttons that trigger the following actions:
    - The Start button calls the **startARODataCollector** method in the *ARODataCollectorLib* library.
    - The Stop button calls the **stopARODataCollector** method in the *ARODataCollectorLib* library.
4. Implements the **raiseARODataCollectorException** method in the *ARODataCollectorLibCallBack* interface to handle exceptions that are sent from the *ARODataCollectorLib* library.
5. Implements the **raiseARODataCollectorCallBack** method in the *ARODataCollectorLibCallBack* interface to handle status notifications sent from the *ARODataCollectorLib* library.

As mentioned in points 4 and 5 of the preceding list, the ARODataCollectorLibTestApp contains two callback methods that have been implemented in order for the calling application to be notified of exceptions and status changes reported by *ARODataCollectorLib*.

The **raiseARODataCollectorException** callback method handles the following exceptions that are reported by *ARODataCollectorLib*:

Exception Name	Explanation
VIDEO_FAILED_START	The video capture has failed to start.
AROTRACE_STOPPED_ERROR	The collector trace has stopped mid trace.
SDCARD_MOUNTED_START	The SD card is either not available or it is mounted to a computer via a USB cable.
SDCARD_MOUNTED_MID	The SD Card has been mounted mid trace and the ARO Data Collector is stopped.
SDCARD_FULL	The SD Card memory is full and the ARO Data Collector is stopped.
NO_NETWORK_CONNECTION	The ARO Data Collector can not start without any active network connection (either Mobile or WiFi).
NO_NETWORK_CONNECTION_MID	The ARO Data Collector has stopped due to the loss of an active network connection (either Mobile or WiFi).
TRACE_INVALID_PATH	The trace folder name cannot have special characters or spaces.
WRITE_FILE_ERROR	An error has occurred while trying to write to one of the ARO Data Collector output files.

The **raiseARODataCollectorCallBack** callback method handles the following status notifications from *ARODataCollectorLib*:

Status Name	Explanation
AROTRACE_START_INPROGRESS	The ARO Trace is attempting to start. It can take up to 15 seconds for the start status to update to either “Completed” or “Failed”.
AROTRACE_START_COMPLETED	The ARO Trace has started successfully and is now recording.
AROTRACE_START_FAILED	The ARO Trace has failed to start.
AROTRACE_STOP_INPROGRESS	The ARO Trace is attempting to stop. Stopping can take up to 30 seconds to complete.
AROTRACE_STOP_COMPLETED	The ARO Trace has stopped. Stop errors are reported back in the Exception Call Back (Note: **raiseARODataCollectorException** is the exception callback method in this example application**)**.

5. Additional Resources

For more information about AT&T ARO, Open Source ARO, and Android Library Projects, see the following resources:

Resource	Description
http://developer.android.com/tools/projects/index.html	Contains more detailed descriptions about Android Library Projects and how to manage them.
http://developer.att.com/ARO	More information about the AT&T ARO application.
https://github.com/attdevsupport/ARO	The github site for Open Source ARO where you can download the ARO code and complete API documentation.