AppDynamics Phonegap Plugin

Introduction

This plugin wraps the AppDynamics SDK for both Andriod and iOS. This provides javascript functions to call the native SDK.

Installation

The plugin is currently not available via the public plugin repository. The plugin is installed in the usual way.

```
cordova plugin add <path/to/plugin>
```

As well as installing the plugin the AppDynamics SDK will need to be installed as normal under the platforms of your Phonegap project.

Android

For Android see https://docs.appdynamics.com/display/PRO40/Instrument+an+Android+Application

Here is an example custom_rules.xml file for Ant.

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
    <target name="-pre-compile">
        <!-- Fix library references due to bug in build.xml: See: https://
groups.google.com/forum/#!topic/android-developers/0ivH-YqCjzg -->
        <pathconvert property="fixedJarsPath" refid="project.all.jars.path">
          <filtermapper>
            <replacestring from="/bin/" to="/ant-build/"/>
            <replacestring from="\bin\" to="\ant-build\"/>
          </filtermapper>
        </pathconvert>
        <path id="project.all.jars.path">
          <pathelement path="${fixedJarsPath}"/>
        </path>
        <echo message="Set jars path to: ${toString:project.all.jars.path}"/>
    </target>
    <!-- Rename AndroidManifest.xml so that Eclipse's import wizard doesn't
detect ant-build as a project -->
    <target name="-post-build">
        <move file="ant-build/AndroidManifest.xml" tofile="ant-build/
AndroidManifest.cordova.xml" failonerror="false" overwrite="true" />
        <move file="CordovaLib/ant-build/AndroidManifest.xml"
tofile="CordovaLib/ant-build/AndroidManifest.cordova.xml" failonerror="false"
overwrite="true" />
```

```
    </target>


    <!-- AppDynamics -->
    <target name="-post-compile">
      <taskdef name="injector"
            classname="com.appdynamics.android.ant.EUMAgentInjectorTask"
            classpath="ADEUMInjector.jar"/>
      <injector classfilespath="${out.classes.absolute.dir}"
            outputlocation="${out.absolute.dir}/instrumented-jars/"
            instrumentationjarlocation="${jar.libs.absolute.dir}/ADEUMAgent.jar"
            jarfilesrefid="project.all.jars.path"
            androidjarlocation="${project.target.android.jar}"/>
    </target>
</project>
```

If you are not using the AppDynamics cloud collector you should use this alternative initialisation code:

```
Instrumentation.start("YOUR KEY", getApplicationContext(), "http://
collector.mycompany.con:7001/", true);
```

Substitute your values in the example above. The final "true" switch is to enable logging; useful for debugging. You can see the logging output with:

```
adb logcat -s appdplugin,ADInstrumentation
```

---

iOS

For iOS see https://docs.appdynamics.com/display/PRO40/Instrument+an+iOS+Application

If you are not using the AppDynamics cloud collector you should use this alternative initialisation code:

```
[ADEumInstrumentation initWithKey:@"YOUR KEY" collectorUrl:@"http://
collector.mycompany.com:7001" enableLogging:true];
```

Substitute your values in the example above. The final "true" switch is to enable logging; useful for debugging.

Usage

The plugin provides a number of javascript functions which may be called from your HTML5 application to access the SDK functionality.

```
<script src="plugins/appdynamics.js"></script>

window.appdynamics.reportMetricWithName(name, value, callback)

window.appdynamics.leaveBreadcrumb(name, callback)

window.appdynamics.startTimerWithName(name, callback)

window.appdynamics.stopTimerWithName(name, callback)
```

```
window.appdynamics.beginCall(name, method, callback)
window.appdynamics.endCall(key, callback)
window.appdynamics.beginHttpRequest(url, callback)
window.appdynamics.reportDone(key, status, headers, callback)
window.appdynamics.consoleLog(msg, callback)
```

The javascript function names map directly to those of the SDK, the plugin takes care of any data type conversion. The callback will be passed a string “Error” if it failed.

---

reportMetricWithName

This function allows you to send name integer pairs to the collector, these will show up under “Custom Metrics” in the AppDynamics dashboard.

```
window.appdynamics.reportMetricWithName(“foo”, 42, function(){});
```

---

leaveBreadcrumb

This function allows you to record the user’s progress through the application. The breadcrumb trail will be included in any crash reports.

```
window.appdynamics.leaveBreadcrumb(“login”, function(){});
```

---

startTimerWithName / stopTimerWithName

This function allows you to time any arbitrary block of code. Remember to call the matching stopTimerWithName.

```
window.appdynamics.startTimeWithName(“process records”, function(){});
// do some processing
window.appdynamics.stopTimerWithName(“process record”, function(){});
```

---

beginCall / endCall

This function allows you to time and count executions of a function. The call to beginCall returns in its callback a unique key which must be used to terminate the matching endCall.

```
function myFunction() {
      var appdKey;
      window.appdynamics.beginCall(“foo”, “bar”, function(key){ appdKey =
key });
      // do some stuff that takes time
      window.appdynamics.endCall(appdKey, function(){});
}
```

beginHttpRequest / reportDone

This function allow you to time network requests (AJAX) calls to the server side. This will track the response time, execution count and status of the requests.

To have full correlation with a server side application also monitored by an AppDynamics application agent a couple of HTTP request headers must be set before making the call.

```
ADRUM = isAjax:true

ADRUM_1 = isMobile:true
```

If these headers are not set in the request then correlation from the mobile application to the server side will not happen.

Here is an example of making and timing a network request using this plugin and the Angularjs framework.

```
var appdKey;

window.appdynamics.beginHttpRequest(requestURL, function(key) {

            appdkey = key;

        });

        // headers() returns headers object

        $http.get(requestURL, {headers: {ADRUM:'isAjax:true',
ADRUM_1:'isMobile:true'}}).success(function(data, status, headers) {

            window.appdynamics.reportDone(appdkey, status, headers(), function()
{});
```

consoleLog

This is a convenience function to enable you javascript code to write log messages to the native application log; useful for debugging.

```
window.appdynamics.consoleLog(“informative message”, function(){});
```