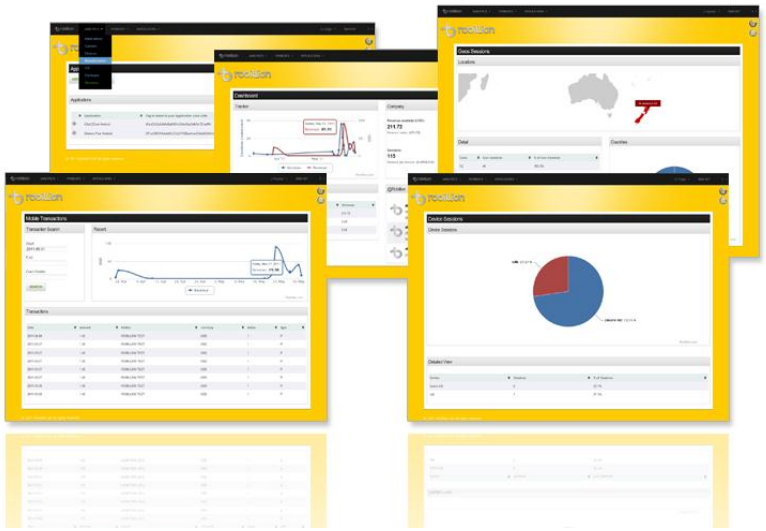
Robillion API

June 21, 2011

Getting started with Robillion for Android

Julian Rayner, Architect, Robillion Ltd



This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Robillion Ltd on the issues discussed as of the date of publication. Because Robillion must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Robillion, and Robillion cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. ROBILLION MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

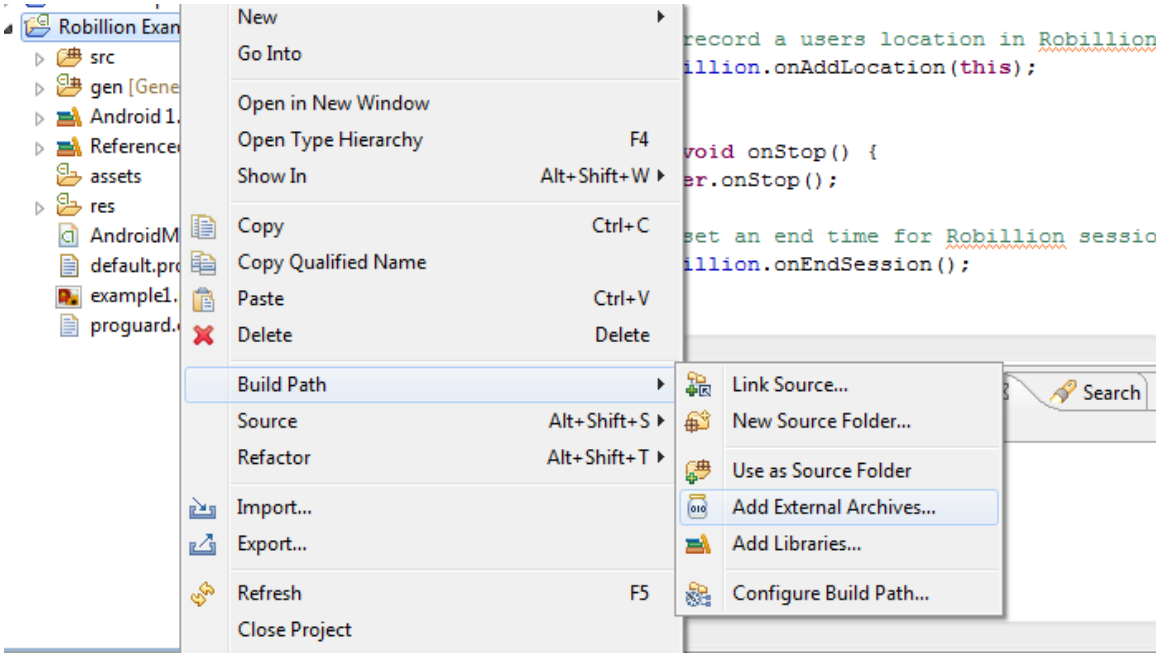
Robillion may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Robillion, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in examples herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

Adding Robillion API

To add Robillion analytic and/or payment capabilities to your application, you will need to add the RobillionApi.jar to your application's classpath: Depending on your build environment there are different steps.

- If you're using Eclipse, modify your Java Build Path, and choose Add External Archivesé



- If you're using the SDK tools directly, drop it into your libs folder and the ant task will pick it up

Adding permissions to AndroidManifest.xml

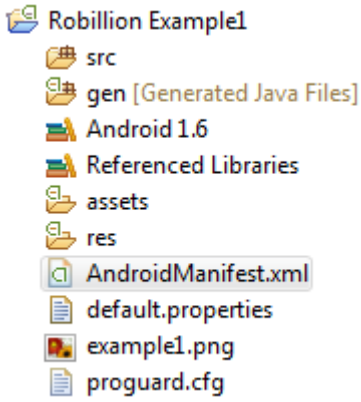
In order for Robillion to send analytic information or request payment processing the internet permission must be set for your project. Optionally, permissions can be set for location analytics.

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

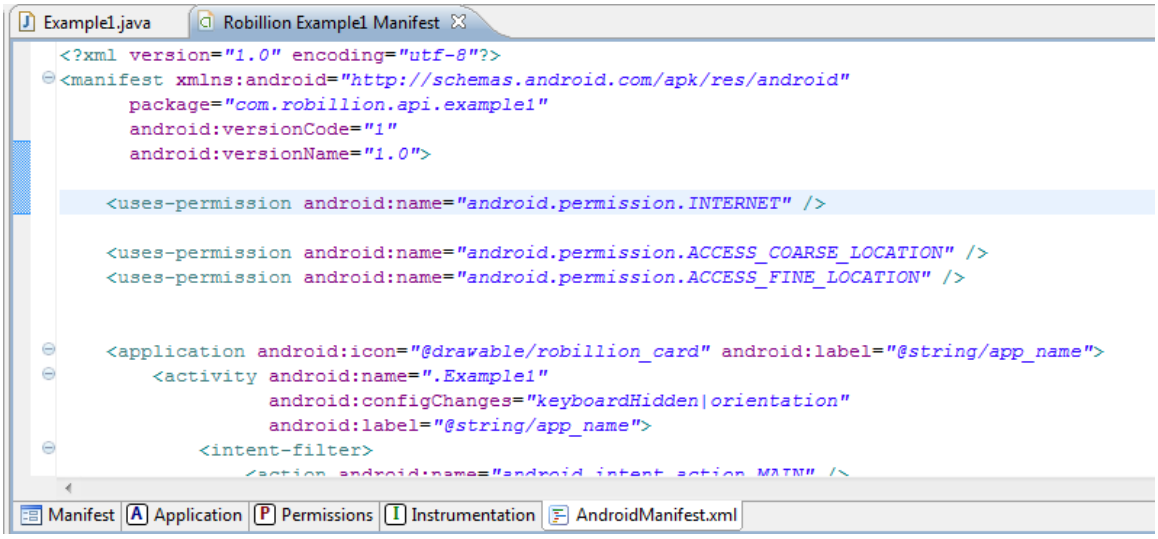
Optional, locationé

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

In eclipse simple double click on the AndroidManifest.xml document in your project tree.



Then add the appropriate permission entries to your xml,



Adding onStartSession and onEndSession

Next we want to add the basic calls to start our Robillion session. Firstly we need a few important variables to define who we are and to give us access to Robillion.

Example1.java

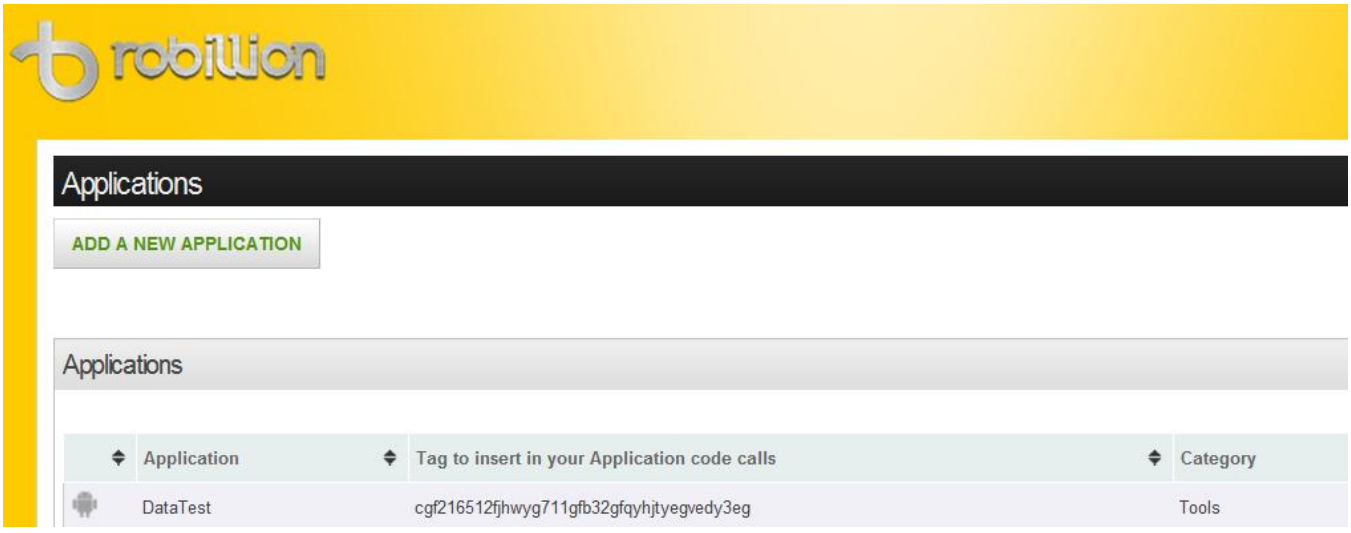
```
package com.robillion.api.example1;

import com.robillion.api.RobillionAgent;

public class Example1 extends Activity {
    public static final String ROBILLION_USER              = "b1b08475ffb08475ffb1b28c5890100b52161f97";
    public static final String ROBILLION_APPLICATION       = "cgf216512fjhwyg711gfb32gfqyhjtyegvedy3eg";

    RobillionAgent robillion = RobillionAgent.getInstance();
```

Add the RobillionAgent instance, and the two string variables. ROBILLION_USER is the 40 character tag you were given when you first joined Robillion, for your company. The ROBILLION_APPLICATION is the tag your application is given if you open your dashboard and navigate to the manage applications you will see this tag. Every application has a unique 40 character application tag, used by the Robillion service to open a user session for a particular application.



Next, in your main/first activity you need to add onStartSession and onEndSession calls. We recommend you add these to the onStart and onStop methods of your activities in your application. Robillion uses a singleton for session scope, this helps removes false session starts from multiple activities being called with onStartSession included.

onStart and onStop methods,

Example1.java

```
    public void onStart() {
        super.onStart();

        /* open a Robillion analytics session, one session is created per lifetime
         * of application or service */
        robillion.onStartSession(this, ROBILLION_USER, ROBILLION_APPLICATION);

        /* record a users location in Robillion analytics */
        robillion.onAddLocation(this);
    }

    public void onStop() {
        super.onStop();

        /* set an end time for Robillion session */
        robillion.onEndSession();
    }
}
```

Adding Location

The more a developer or marketer knows about their market the better the software company can target new features, promotions and support. Robillion has a number of extra analytics on the way, this current build includes Location recording.

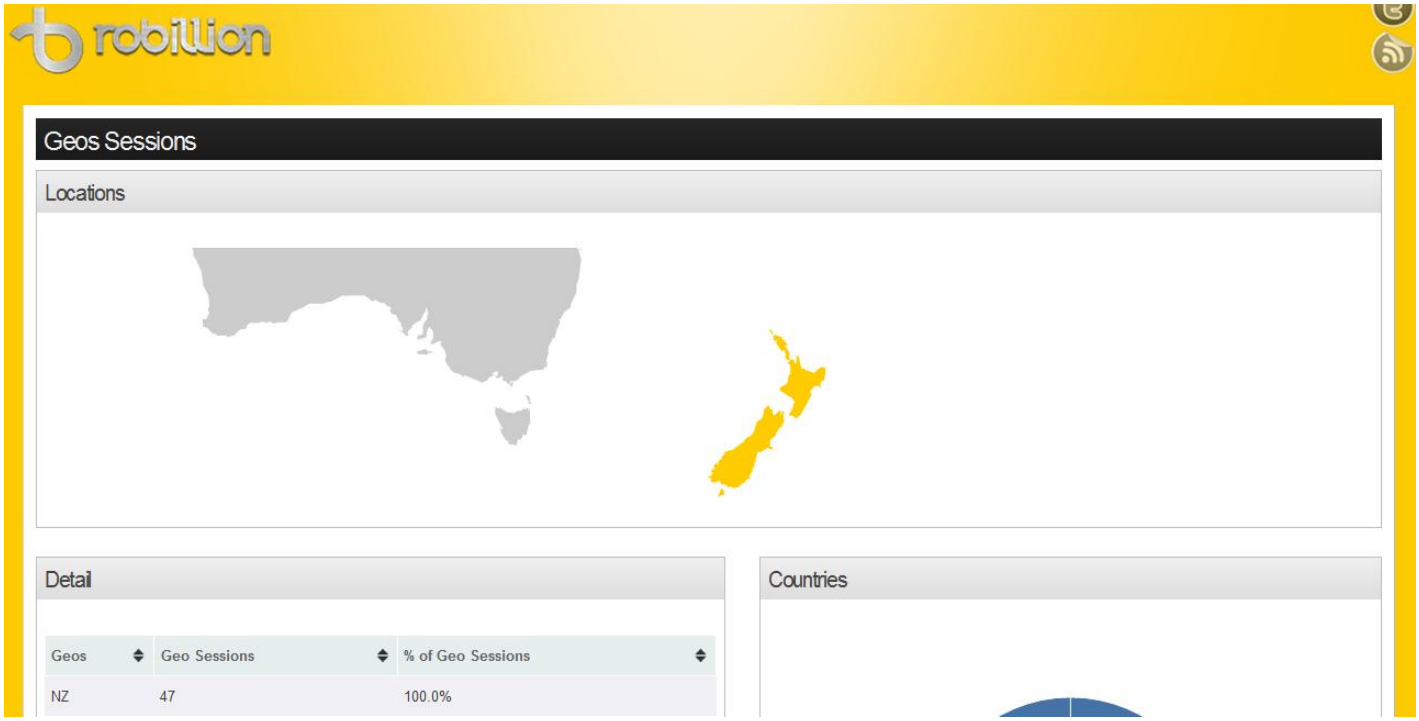
Simple add one extra line robillion.addLocation(this), to your activities onStart directly after the onStartSession call and your done.

```
public void onStart() {
    super.onStart();

    /* open a Robillion analytics session, one session is
     * of application or service */
    robillion.onStartSession(this, ROBILLION_USER, ROBILI

    /* record a users location in Robillion analytics */
    robillion.onAddLocation(this);
}
```

You will now see location analytics in your dashboard,



Adding Payments

Due to the asynchronous nature of a credit card/payment system. Adding payments requires a little more in-depth operation handling. Luckily we have added easy to use functions to encapsulate this for you, but still ensure you can skin the complete payment system in keeping with your particular app/service.

For a minimal solution, as more complex UIôs is out of scope for this document, but available in the open source examples on our developer resources page.

```
/* tell Robillion to process payment */
RobillionAgent robillion = RobillionAgent.getInstance();
robillion.onPayment(cardHolder,
                    cardNumber,
                    _transactionAmount,
                    _currency,
                    cardExpiry,
                    cardVerification);
```

Robillion also provides methods to check validity of your cardHolder, cardNumber, cardExpiry and cardVerification strings.

```
private boolean isFormValid(String cardNumber, String cardExpiry, String cardHolder, String cardVerification) {
    RobillionAgent robillion = RobillionAgent.getInstance();

    if(!robillion.isValidCardNumber(cardNumber)) {
        Toast.makeText(PaymentDialog.this.getContext(),
                "Invalid Credit Card Number",
                Toast.LENGTH_LONG).show();
        return false;
    }

    if(!robillion.isValidExpiration(cardExpiry)) {
        Toast.makeText(PaymentDialog.this.getContext(),
                "Invalid Expiry Date",
                Toast.LENGTH_LONG).show();
        return false;
    }

    if(!robillion.isValidCardHolder(cardHolder)) {
        Toast.makeText(PaymentDialog.this.getContext(),
                "Invalid Card Holder Name",
                Toast.LENGTH_LONG).show();
        return false;
    }

    if(!robillion.isValidCvv2(cardVerification)) {
        Toast.makeText(PaymentDialog.this.getContext(),
                "Invalid Card Verification Number",
                Toast.LENGTH_LONG).show();
        return false;
    }

    return true;
}
```

Currency 3 character string must be from the following list, or Robillion will default to US dollars (USD).

ISO currency code - NZD, AUD, FJD, USD etc.

CAD	Canadian Dollar
CHF	Swiss Franc
EUR	Euro
FRF	French Franc
GBP	United Kingdom Pound
HKD	Hong Kong Dollar
JPY	Japanese Yen
NZD	New Zealand Dollar
SGD	Singapore Dollar
USD	United States Dollar
ZAR	Rand
AUD	Australian Dollar
WST	Samoan Tala
VUV	Vanuatu Vatu
TOP	Tongan Pa'anga
SBD	Solomon Islands Dollar
PNG	Papua New Guinea Kina
MYR	Malaysian Ringgit
KWD	Kuwaiti Dinar
FJD	Fiji Dollar

Since it may take a short period for Transactions to process it is important to register a payment observer with Robillion. The observer is then notified once the payment is processed or in the event of an authorization failure.

```
/* add an observer, so we can react to payment result messages */
RobillionAgent robillion = RobillionAgent.getInstance();
robillion.addPaymentObserver(new PaymentObserver());
```

Make sure you register the payment observer before calling onPayment.

```
public class PaymentObserver implements Observer {
    @Override
    public void update(Observable observable, Object data) {
        ContentValues cv = (ContentValues) data;
//      /* lets simply dump the response from Robillion to log */
//      Set<Entry<String, Object>> s=cv.valueSet();
//      for (Entry<String, Object> entry : s) {
//          Log.d("robillion", entry.getKey() + " := " + entry.getValue().toString());
//      }

        if(progressDialog != null)
            progressDialog.dismiss();

        /* right, what was the result? */
        if(cv.containsKey("Authorized")) {
            Integer paymentResult = cv.getAsInteger("Authorized");
            if(paymentResult>0) {
                /* Tell calling activity we have been paid, and close */
                PaymentDialog.this.dismiss();
            } else {
                /* if payment wasn't a success, let user know and why */
                Log.d("robillion", "error: " + cv.getAsString("Message"));
            }
        }
        else {
            if(cv.containsKey("Message")) {
                Log.d("robillion", "error: " + cv.getAsString("Message"));
            } else {
                Log.d("robillion", "error: Payment failed, unknown error");
            }
        }
    }
}
```

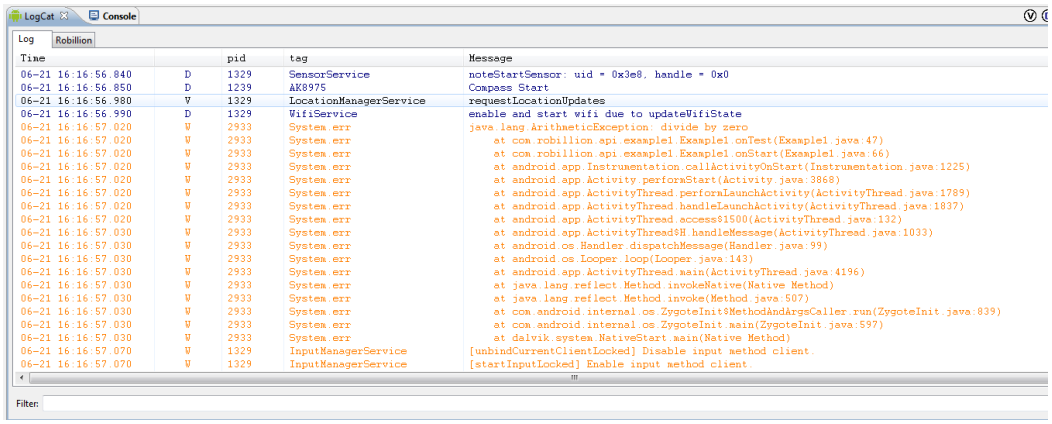
Adding Issue Management

It is often a huge hassle to get errors from your apps/customers to flow through to your testing, support and developers. The Robillion framework makes this easy. One line of code in your exception catches sends issue tracking info to your issue/project management system.

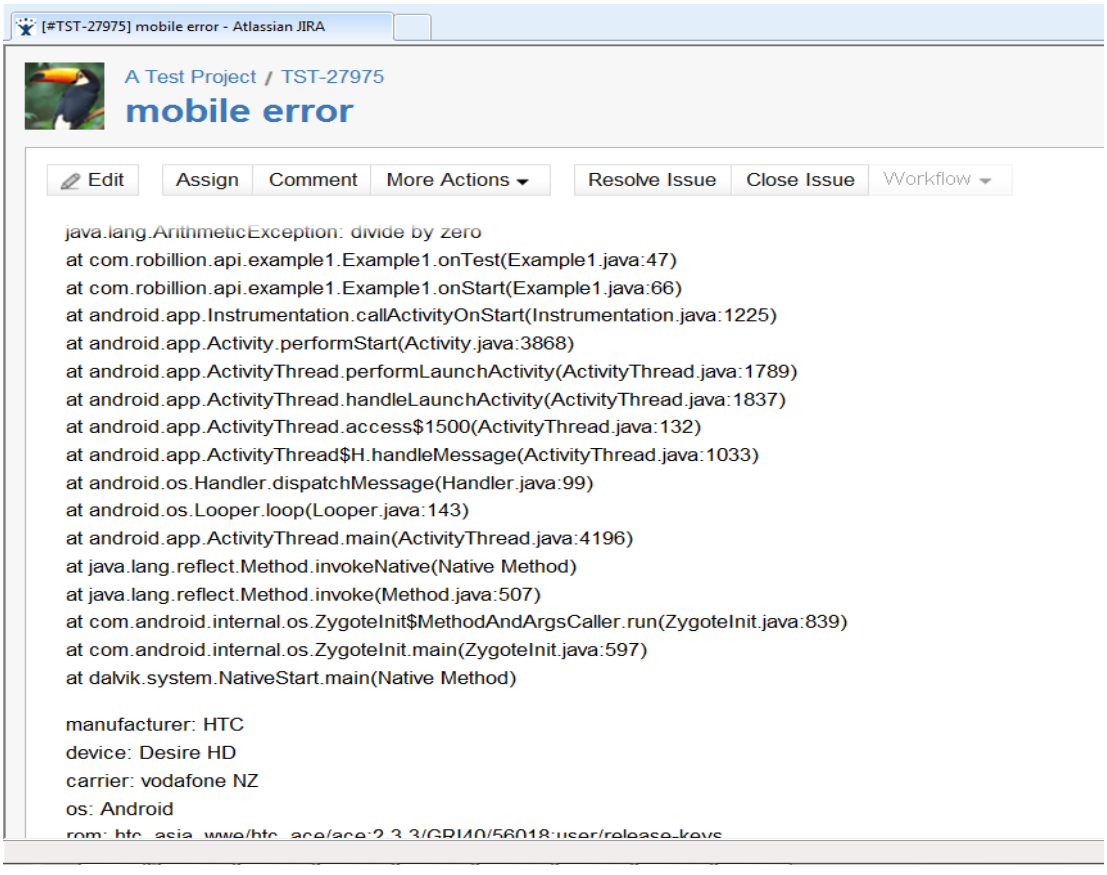
Currently JIRA Project Management by Atlassian and Email are support. We are actively working with project management companies to build connectors. Contact Robillion support to get your mechanism included.

```
public void onTest() {
    try {
        int t = 5/0;
        t=t+1;
    }
    catch(Exception e) {
        robillion.onBug(Example1.this,
                        "TST",
                        "1", "mobile error",
                        e.getMessage() + "\n" + robillion.getStackTrace(e));
        e.printStackTrace();
    }
}
```

Adding the onBug method from Robillion sends the error automatically



To your project management system,



You do need to add the connector details, for errors to automatically flow through. Simply login to Robillion and set your preferred Issue/Project management suite