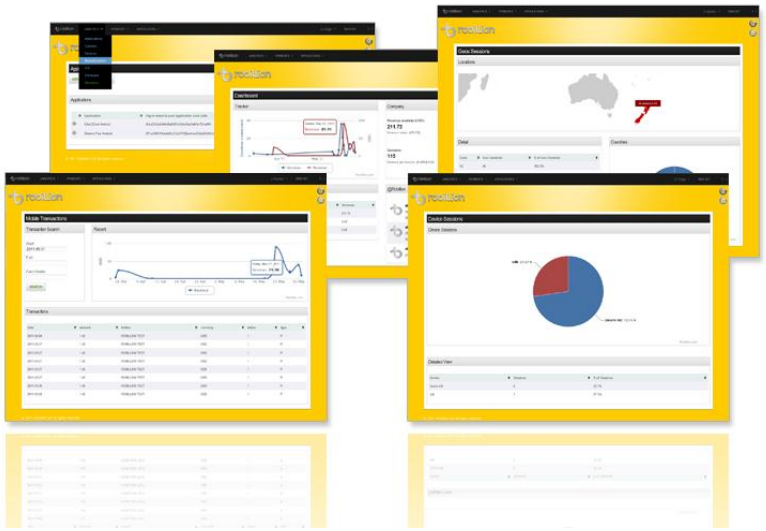
Robillion API

June 21, 2011

Getting started with Robillion for Blackberry

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.

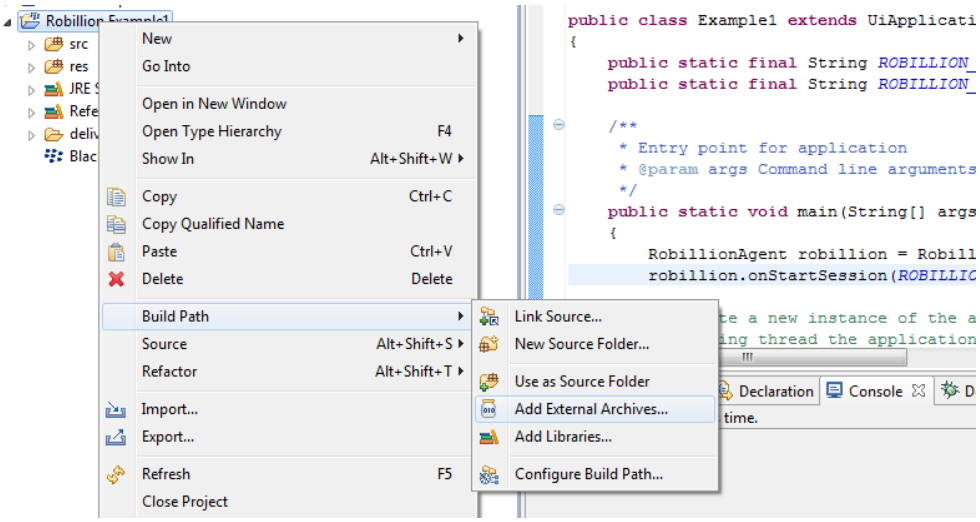
Robillion may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Robillion, the furnishing of this document does not give you any license to these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

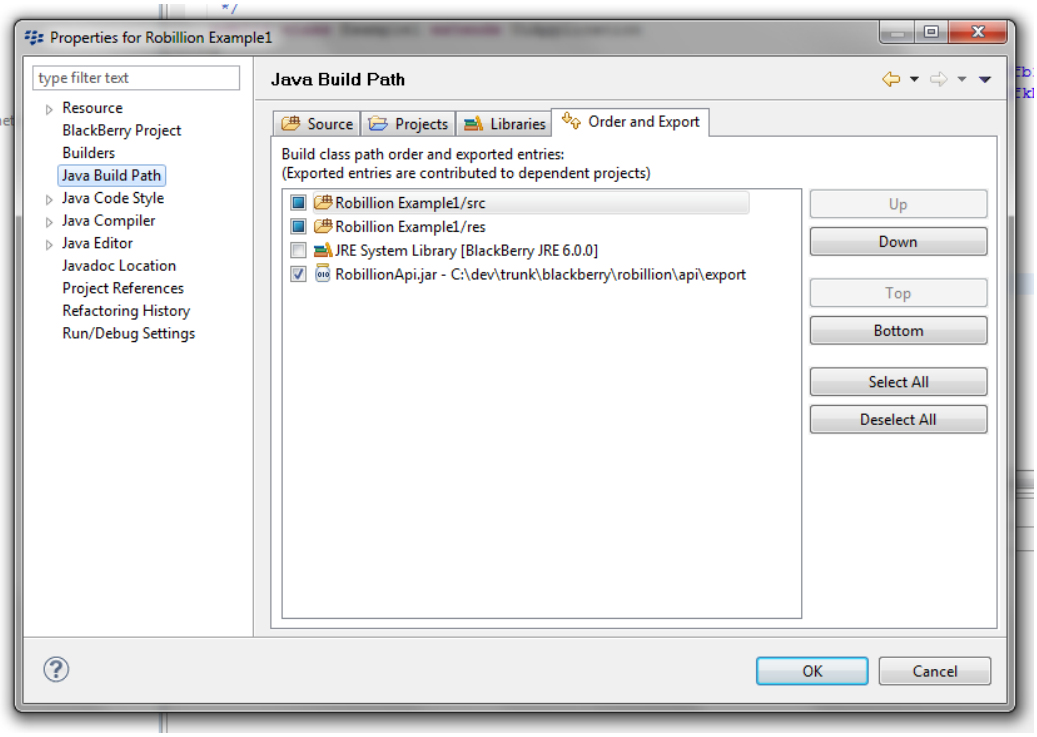
Adding Robillion API

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

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



Remember to check the tick box in your project properties, otherwise youôll get a load module error for your app when it runs.



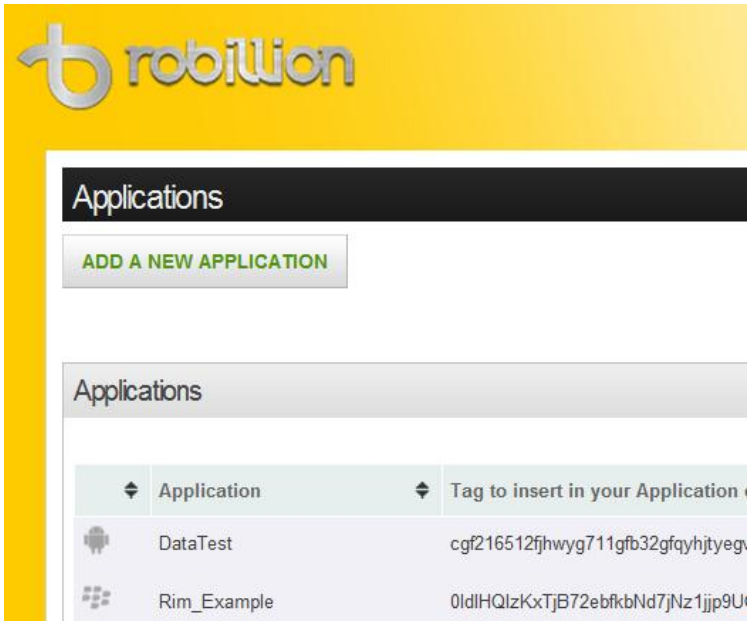
Adding onStartSession and onEndSession

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

```
/**
 * This class extends the UiApplication class, providing a
 * graphical user interface.
 */
public class Example1 extends UiApplication
{
    public static final String ROBILLION_USER           = "b1b08475ffb08475ffb1b28c5890100b5
    public static final String ROBILLION_APPLICATION    = "0IdlHQIzKxTjB72ebfkbNd7jNz1jjp9UQ
```

Add 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 `public static void main(String[] args)` you need to add our onStartSession call. Robillion uses a singleton for session scope, so you can add onStartSession and onEndSession commands in a few places without issue.

onStartSession method in your main,

```
    /**
     * Entry point for application
     * @param args Command line arguments (not used)
     */
    public static void main(String[] args)
    {
        RobillionAgent robillion = RobillionAgent.getInstance();
        robillion.onStartSession(ROBILLION_USER, ROBILLION_APPLICATION);

        // Create a new instance of the application and make the currently
        // running thread the application's event dispatch thread.
        Example1 theApp = new Example1();
        theApp.enterEventDispatcher();
    }
```

onEndSession method in your screen close,

Example1.java

```
    /**
     * Displays a dialog box to the user with the text "Goodbye!" when the
     * application is closed.
     *
     * @see net.rim.device.api.ui.Screen#close()
     */
    public void close()
    {
        // Display a farewell message before closing the application
        Dialog.alert("Goodbye!");

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

        super.close();
    }
}
```

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.

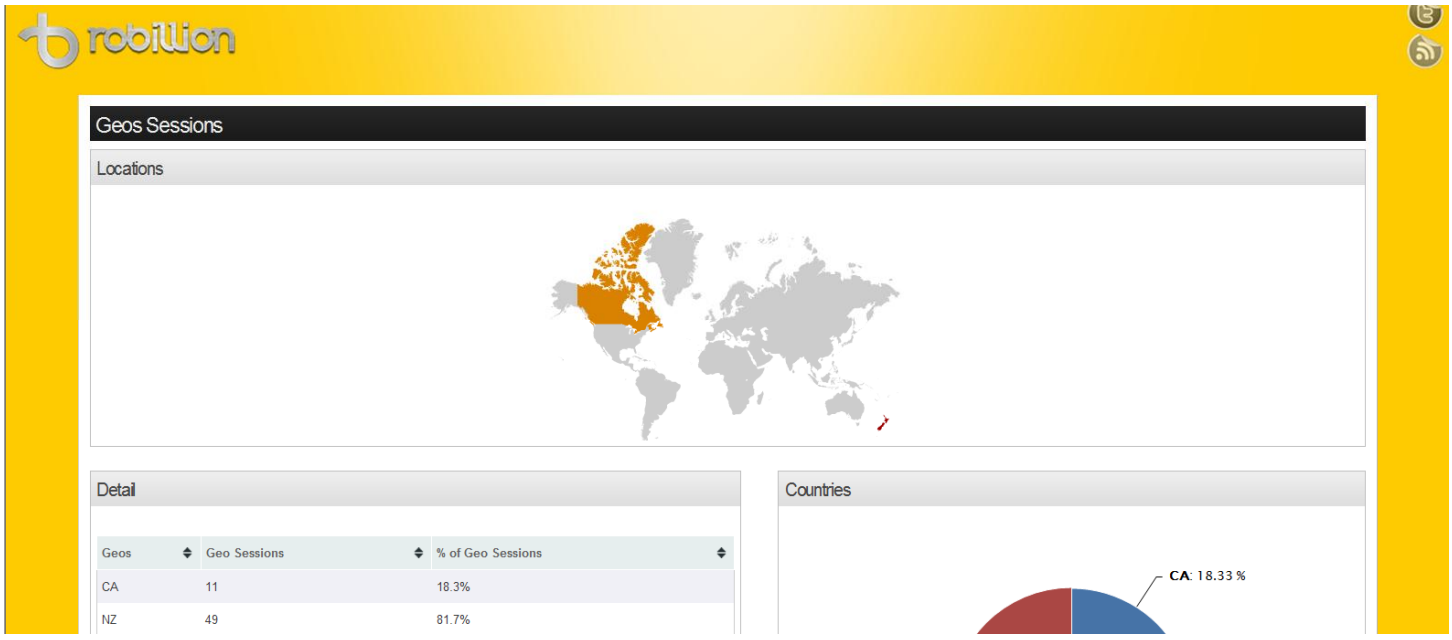
Simple add one extra line robillion.addLocation(this), to your activities Main directly after the onStartSession call and your done.

Example1.java

```
/**
 * Entry point for application
 * @param args Command line arguments (not used)
 */
public static void main(String[] args)
{
    RobillionAgent robillion = RobillionAgent.getInstance();
    robillion.onStartSession(ROBILLION_USER, ROBILLION_APPLICATION);
    robillion.onAddLocation();

    // Create a new instance of the application and make the currently
    // running thread the application's event dispatch thread.
    Example1 theApp = new Example1();
    theApp.enterEventDispatcher();
}
```

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);
```

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.

```
RobillionAgent robillion = RobillionAgent.getInstance();
robillion.addPaymentObserver(new PaymentObserver());
```

Make sure you register the payment observer before calling onPayment.

```
public class PaymentObserver implements Observer {
    public void update(Observable observable, Object data) {
        MultiMap cv = (MultiMap) data;
        for (Enumeration e = cv.elements(); e.hasMoreElements();) {
            Log.d("robillion", "PO: " + e.nextElement().toString());
        }
    }
}
```

Blackberry Connections

Due to some Blackberry connection methods requiring additional parameters attached to your url, we have provided a method to override the default Robillion has automatically determined for network connection. To provide your parameters, make sure you call addURLExtras before onStartSession.

```
/**
 * Entry point for application
 * @param args Command line arguments (not used)
 */
public static void main(String[] args)
{
    RobillionAgent robillion = RobillionAgent.getInstance();
    robillion.addURLExtras(";deviceSide=true");
    robillion.onStartSession(ROBILLION_USER, ROBILLION_APPLICATION);
    robillion.onAddLocation();

    // Create a new instance of the application and make the currently
    // running thread the application's event dispatch thread.
    Example1 theApp = new Example1();
    theApp.enterEventDispatcher();
}
```

Examples;

```
* MDS (to force use of an MDS on the BES) = ";deviceside=false"

* simulators by default takes the above parameter and connects.

* WiFi = ";deviceside=true;ConnectionUID=S TCP-
WiFi;ConnectionSetup=delayed;retrynocontext=true" or simply
";deviceside=true;interface=wifi"

* BIS-B = ";deviceside=false;ConnectionType=mds-public"

* TCP (requiring a correct WAP configuration)= ";deviceside=true"
```

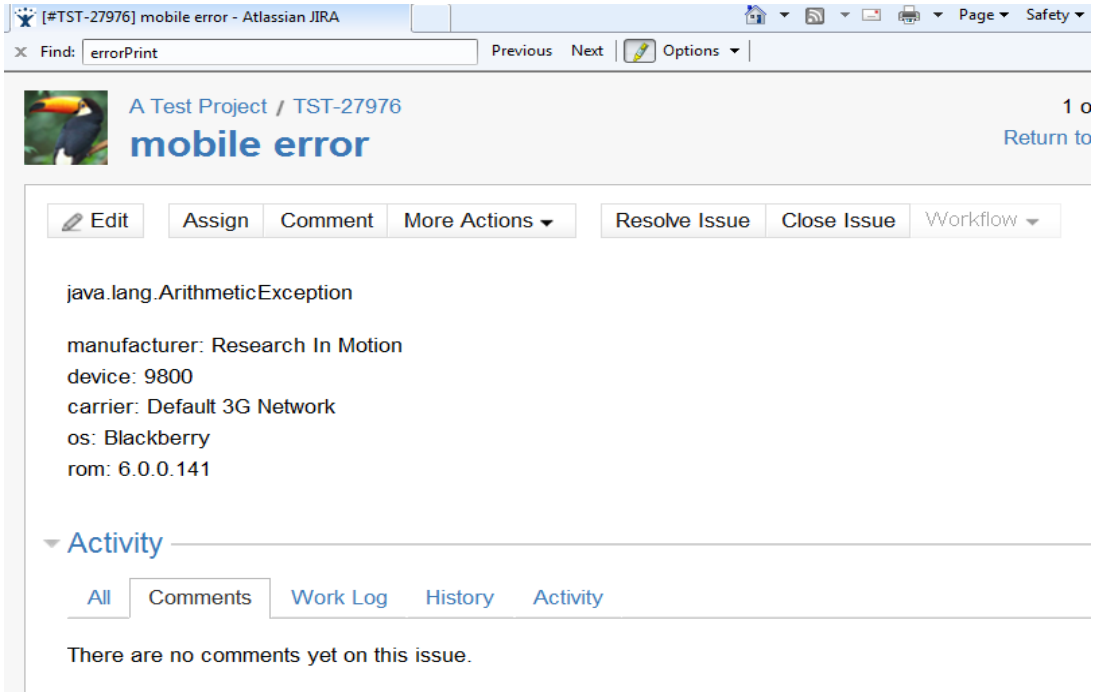
Adding Issue Management

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) {
        RobillionAgent robillion = RobillionAgent.getInstance();
        robillion.onBug("TST",
                        "1", "mobile error",
                        e.getMessage() + "\n" + e.toString());
        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