Wikiprint Book

Title: Test Generation Via Symbolic Simulation

Subject: Java Path Finder - projects/jpf-mango/workshop2012

Version: 21

Date: 02/22/2013 02:40:26 AM

Table of Contents

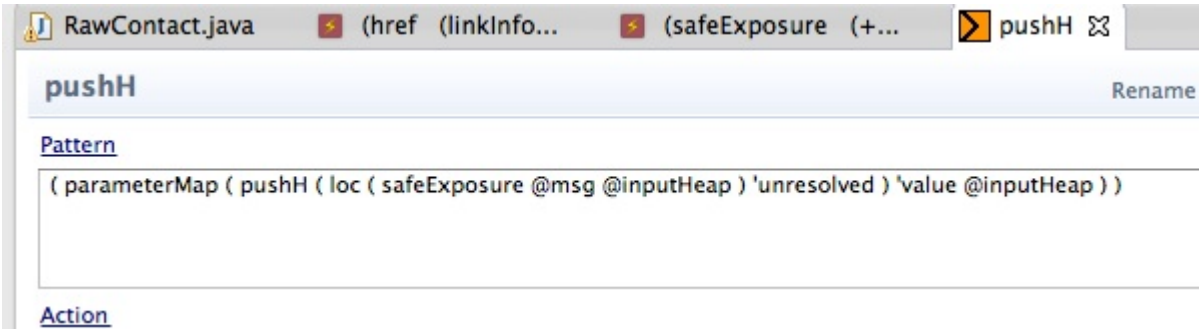
Test Generation Via Symbolic Simulation **3**

Test Generation Via Symbolic Simulation

This demo explains how to train Mango to recognize a potential vulnerability, the inadvertent disclosure of confidential information via a thrown exception message. The tool has already been trained not to "trust" the Log.i() method in RawContact.EZtest() (see figure below).

```
29  */
30  final public class RawContact {
31
32      // DEMO EXAMPLE
33      public JSONObject EZtest() {
34          JSONObject json = new JSONObject();
35
36          try {
37              if (mDeleted) {
38                  json.put("d", mDeleted);
39              }
40          } catch (final Exception ex) {
41              Log.i(TAG, "Error converting RawContact to JSONObject" + ex.toString());
42          }
43
44          return json;
45      }
46      // END DEMO EXAMPLE
47
```

This means that input to Log.i() will be flagged with unknown exposure status. More specifically, the figure below shows the expression for the output heap of the Log.i() specification has been altered by introducing the "safeExposure" sub-expression (see figure below).



The goal is to teach the tool that the message from an exception thrown by an "opaque" method such as put() in the example is in fact unsafe.

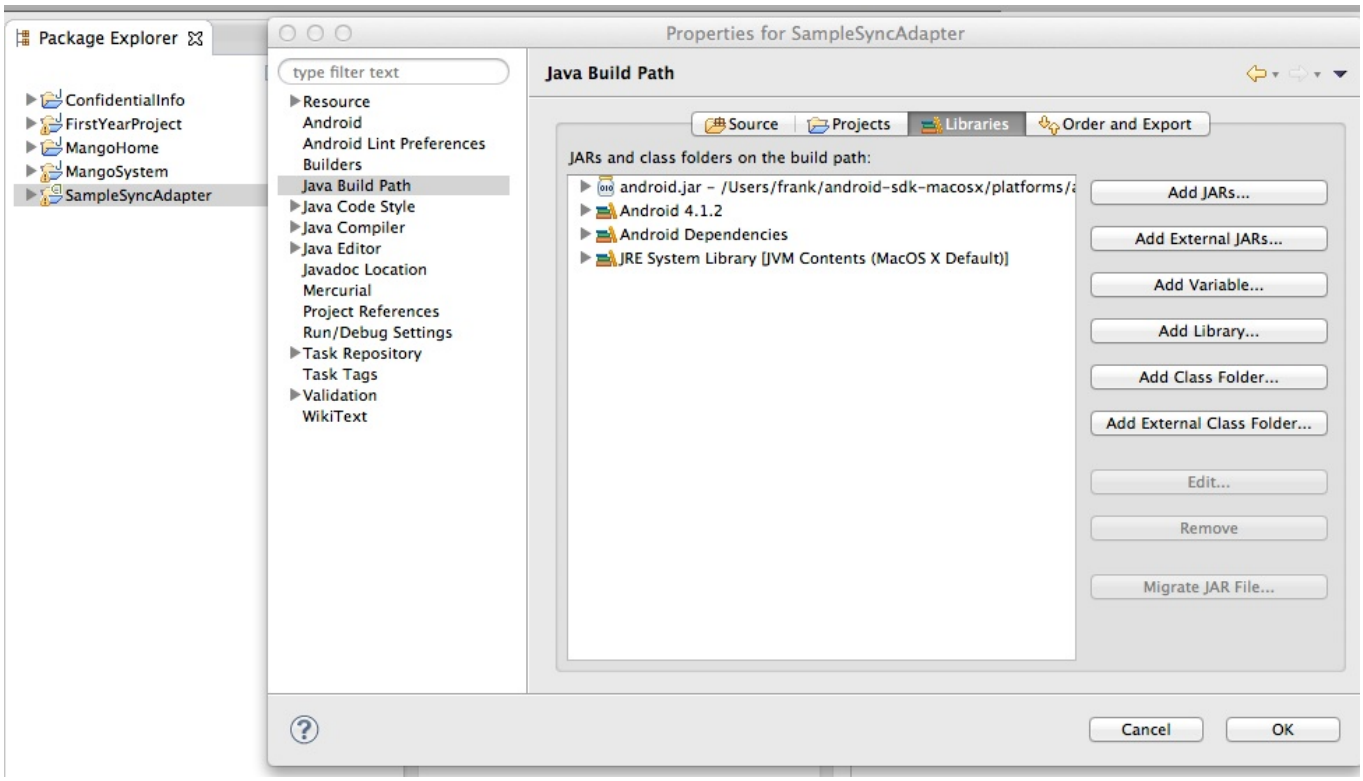
1 Follow the instructions here to install the Mango development platform.

2 Launch the Mango platform from the development platform to create the corresponding Mango run-time, if necessary. You may wish to run the Hello World example from the Mango platform to gain familiarity with base operation of the plugin.

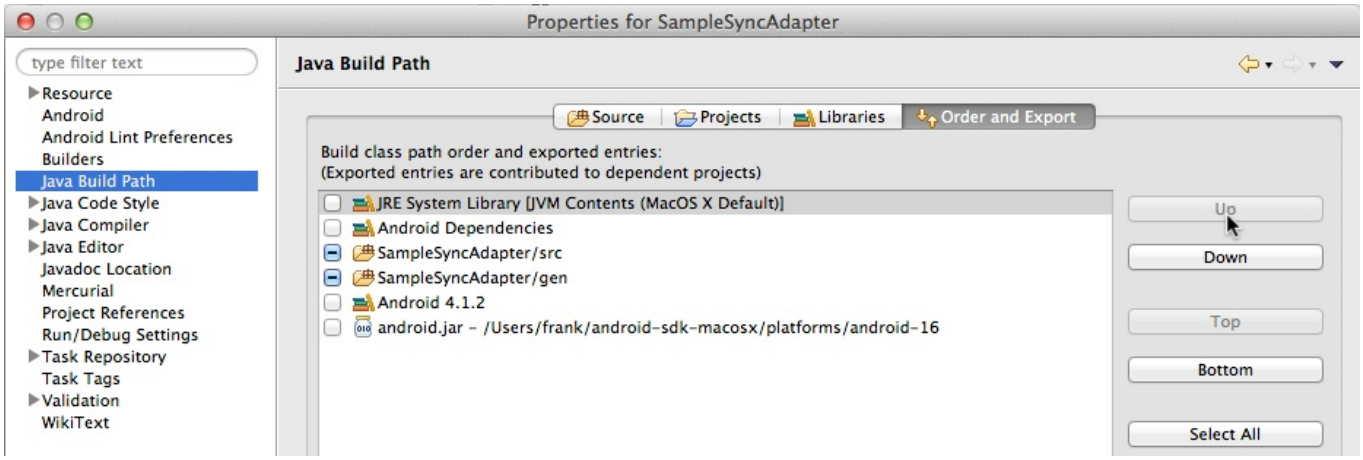
3 Follow the instructions at the Android website http://developer.android.com/sdk/index.html to install and build the Android SDK. In the developer Ecipse platform, load the Eclipse Android plugin, found at the update site:

https://dl-ssl.google.com/android/eclipse/

4 Launch the Mango platform, and create the Android/Example/SampleSyncAdapter project within the Mango platform. You will need to explicitly indicate the android.jar from the Android SDK in the project class path (see figure below). Likewise, explicitly indicate the JRE (Java Runtime Environment) in the project class path.



5 Be sure the JRE is first in the class path order list (see figure below).



6 Download SampleSyncAdapter.zip at http://babelfish.arc.nasa.gov/trac/jpf/attachment/wiki/projects/jpf-mango/SampleSyncAdapter.zip. Drag the MangoHome and MangoSystem found in the unzipped download folder to the Mango platform runtime. If these folders replace the corresponding existing folders in the runtime, refresh the corresponding projects in the Mango platform. The android.jar must be present in the MangoSystem class path. If this link is broken, repair it in the MangoSystem class path preferences. Point to the corresponding location in your Android SDK, as is done in step 4.

7 Be sure the Mango preferences for the Mango platform are set to the default state.

8 Paste the following code into the class com.example.android.samplesync.client.RawContact, located in the SampleSyncAdapter project. Fire the "populate" and "run" commands for the EZtest method.

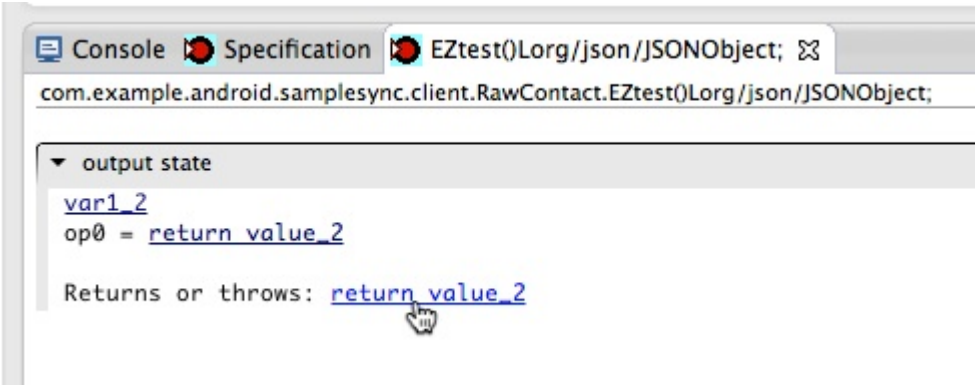
```
   // DEMO EXAMPLE
  public JSONObject EZtest() {
      JSONObject json = new JSONObject();

      try {
          if (mDeleted) {
              json.put("d", mDeleted);
```

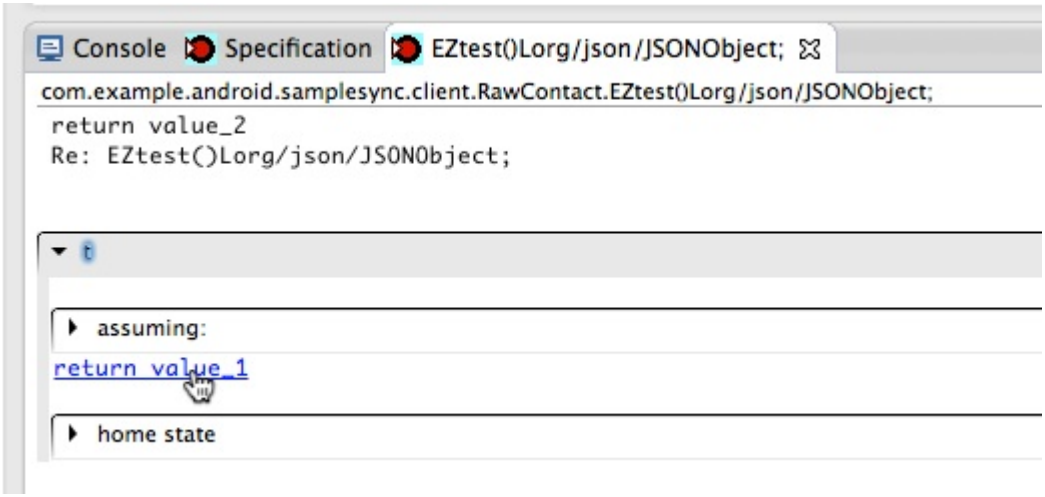
```
            }
        } catch (final Exception ex) {
            Log.i(TAG, "Error converting RawContact to JSONObject" + ex.toString());
        }

        return json;
    }
    // END DEMO EXAMPLE
```

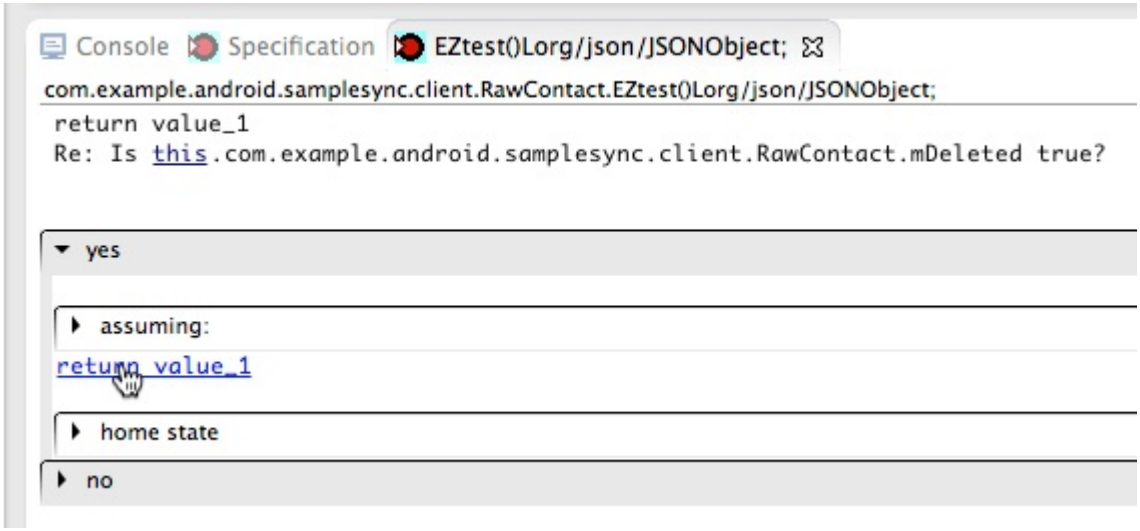
9 View the EZtest formal specification (click on EZtest method icon --> click on EZtest in popup --> double-click on green Tr in Mango Explorer). Click on return_value_2 (see fig below).



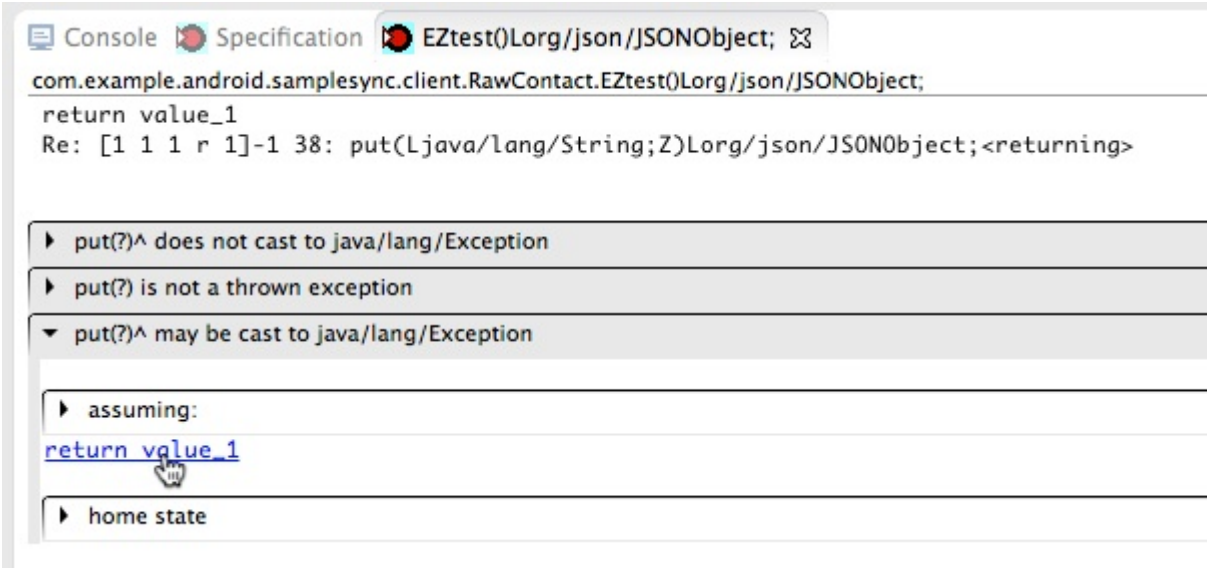
10 Click on return_value_1 (see fig below)



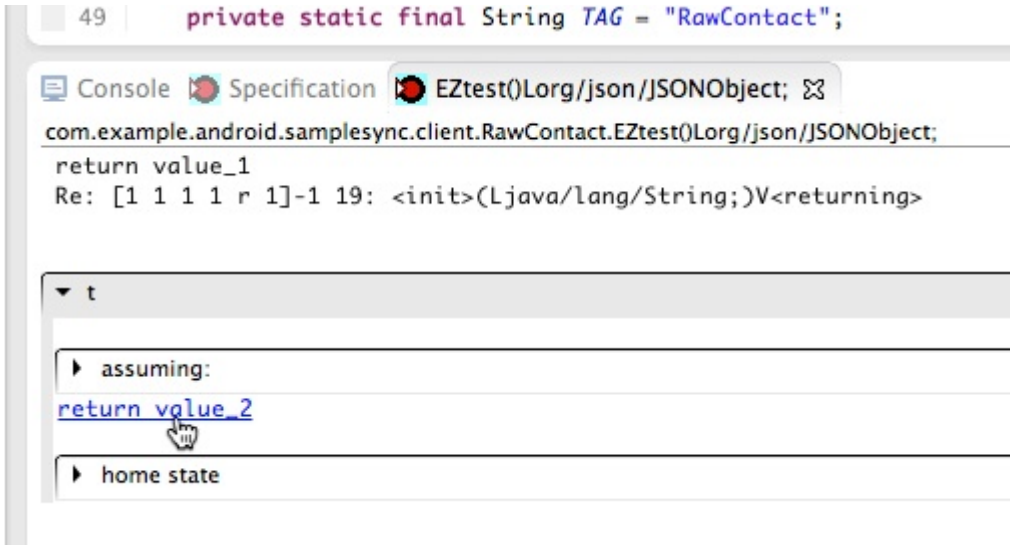
11 Click on return_value_1 (see fig below). Be sure you are in the "yes" case, as this case potentially throws an exception to be caught and exposed by Log.i().



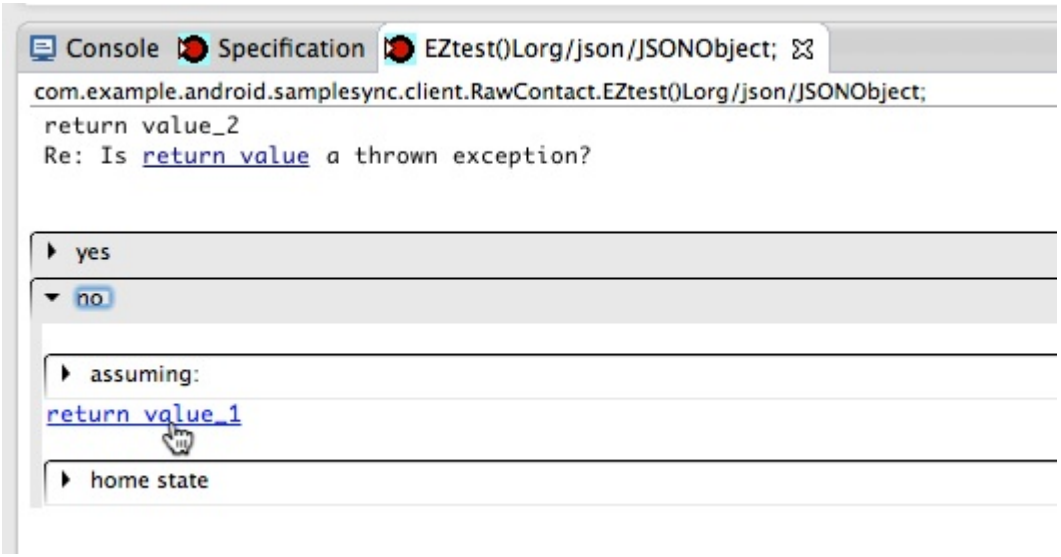
12 Click on return_value_1 in the case `put(?)^ may be cast to java/lang/Exception` (see fig below). Observe this case must succeed in order for the exception handler to capture the thrown exception.



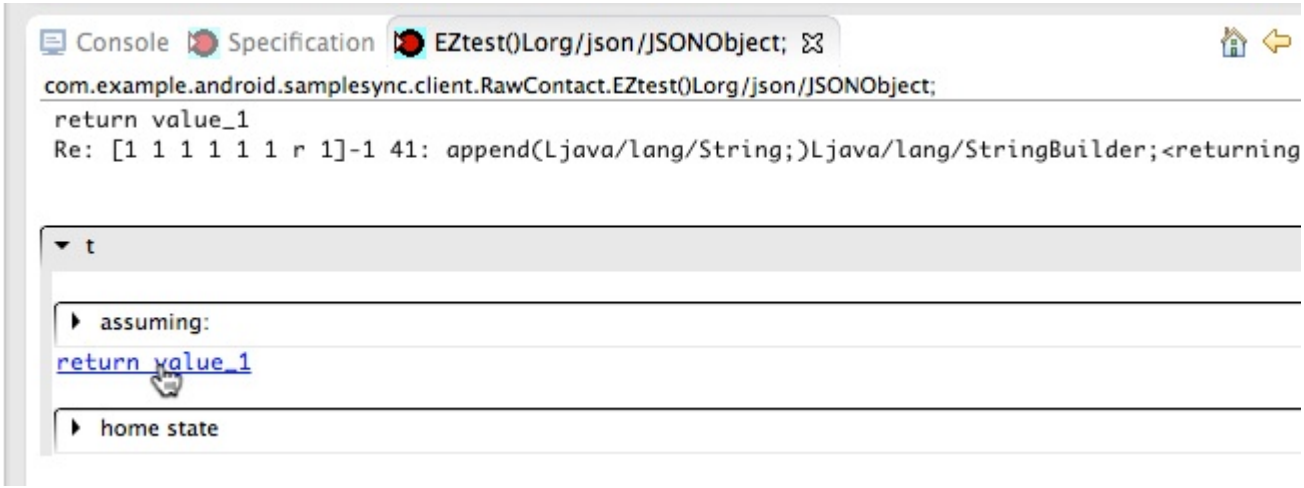
13 Click on return_value_2 (see fig below)



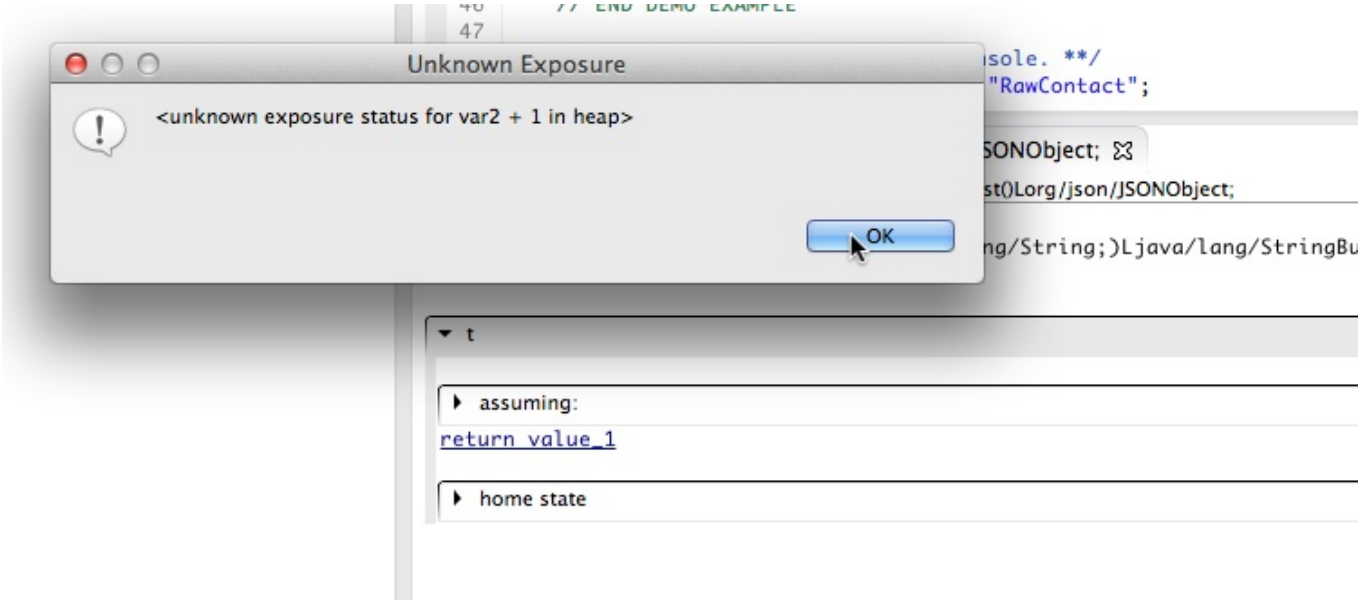
14 Click on return_value_1 in the "no" case (see fig below). The exception in question here is for the concatenation operation in the second argument to Log.i(). If we throw here, the exception message corresponding to json.put() will not be revealed.



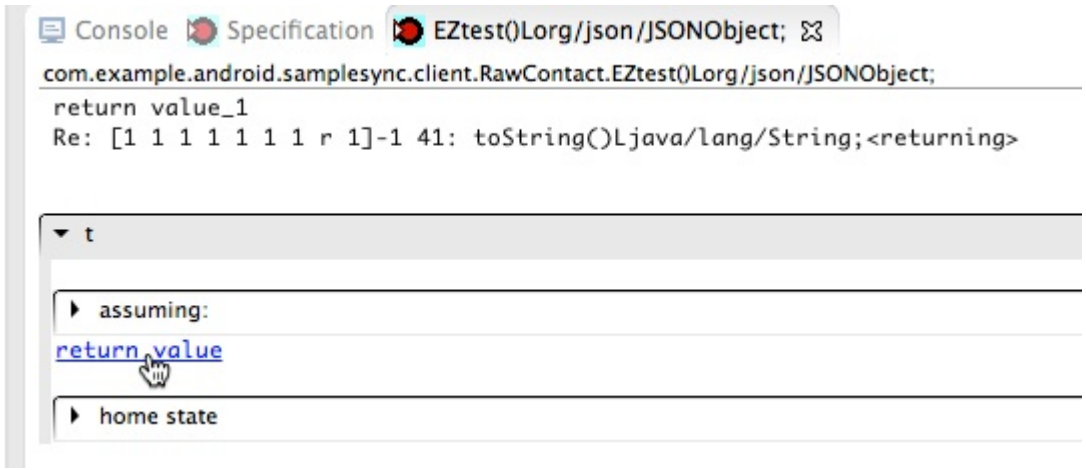
15 Click on return_value_1 (see fig below)



16 Acknowledge the detection of an "unknown exposure". This warning will appear occasionally from now on. In each case just acknowledge the warning.

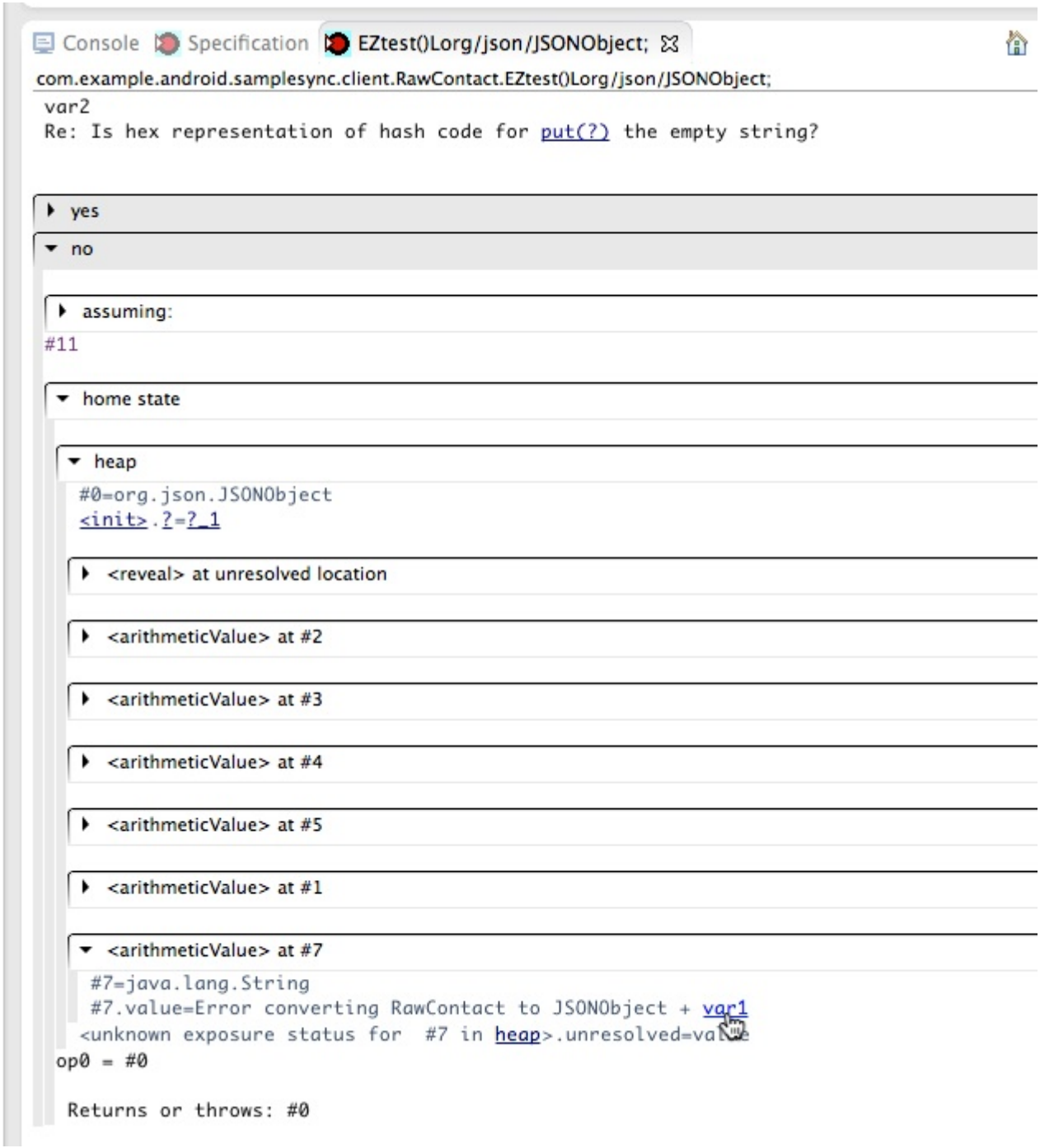


17 Click on return_value (see fig below)

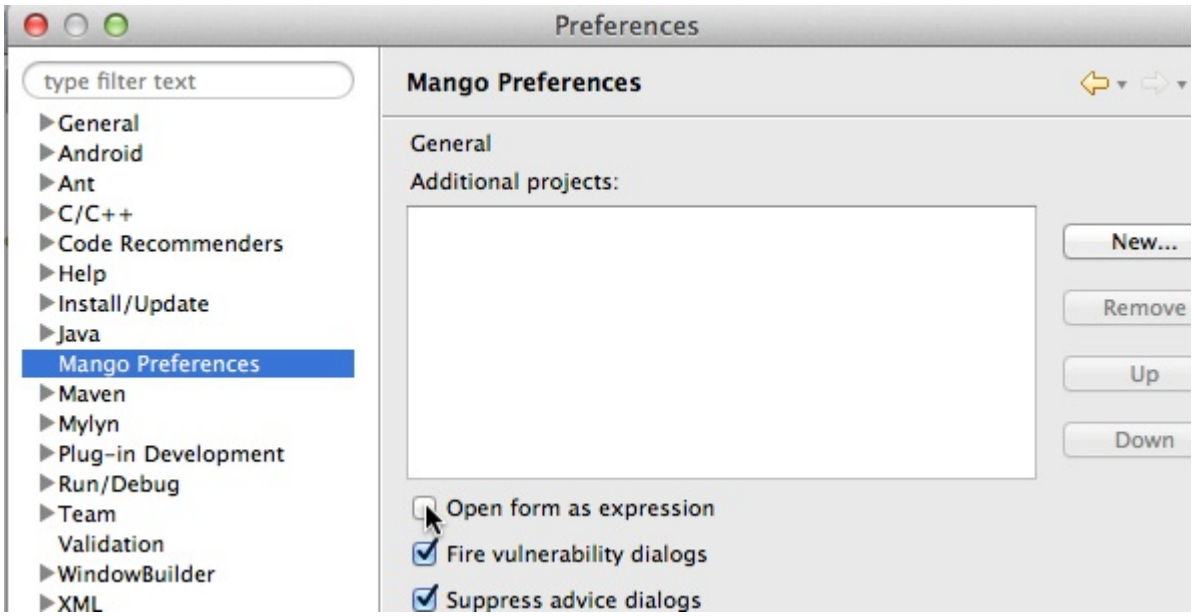


18 Click on var2 in the "no" case (see fig below). If Log.i() throws the put() exception message will not be revealed.

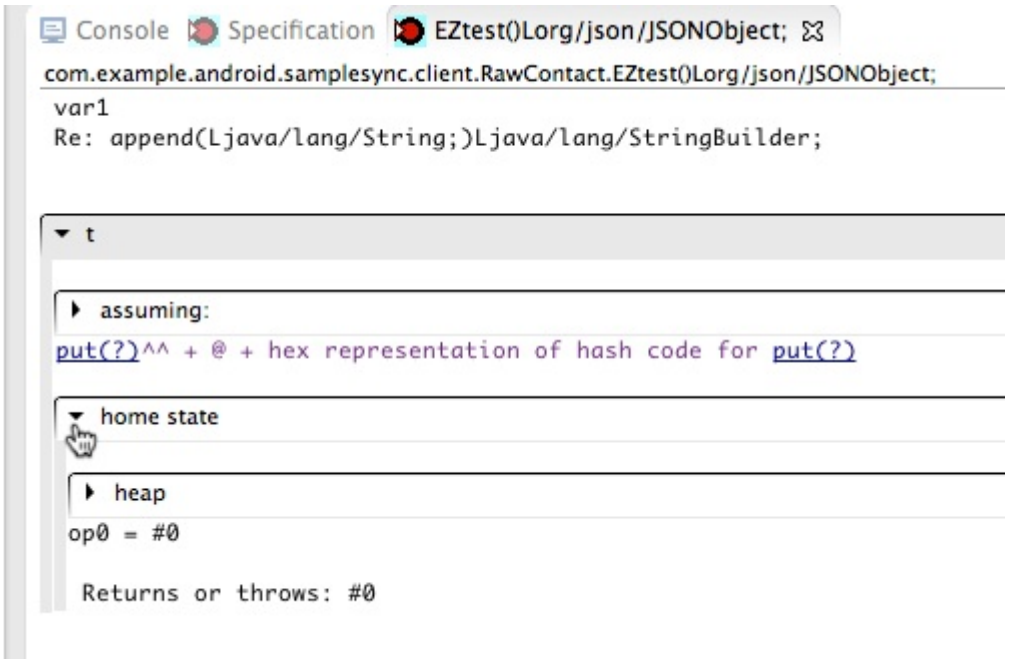
19 Click on var1 in the "no" case (see fig below). To find var1, open the "home state", open the "heap", and finally open heap item #7. Note in this item that "Error converting RawContact to JSONObject" is a string literal which occurs in the code. Evidently var1 is the message for the exception thrown by put().



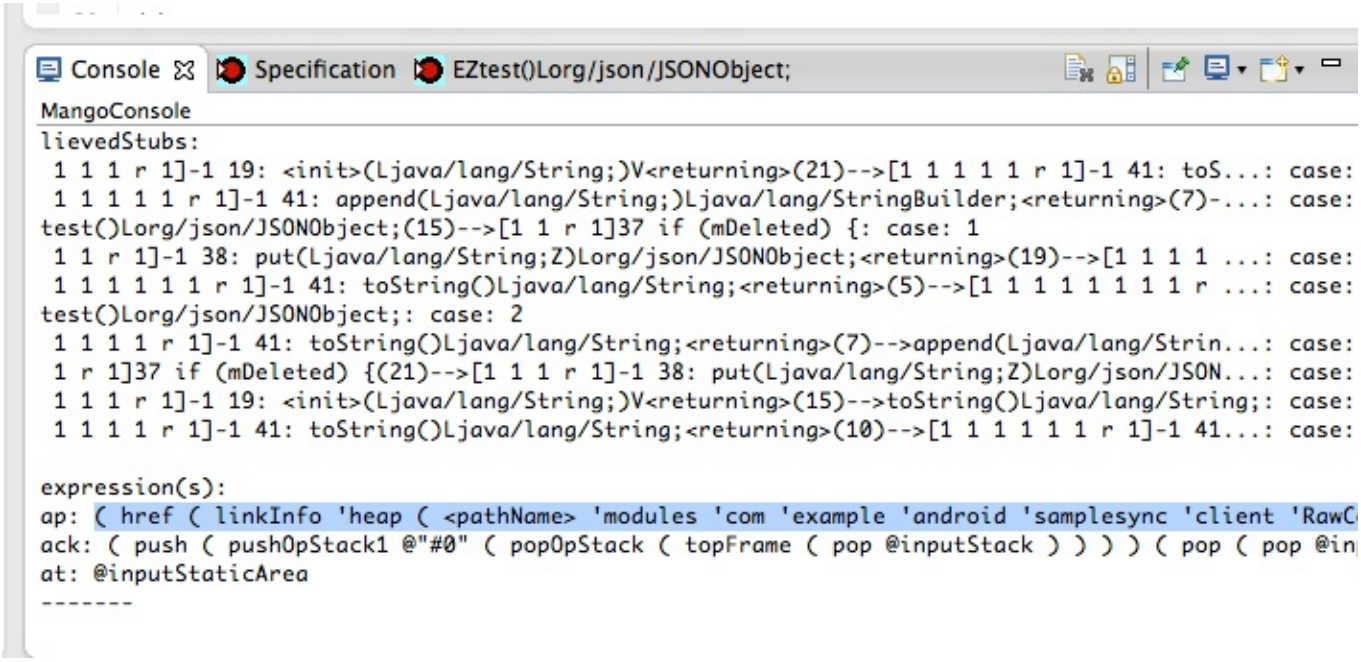
20 At this point, it is necessary to reveal the "home state" in the Console. In order to do this, open the Mango Preferences and check "Open form as expression". Normally this is left unchecked because of the load it puts on the console.



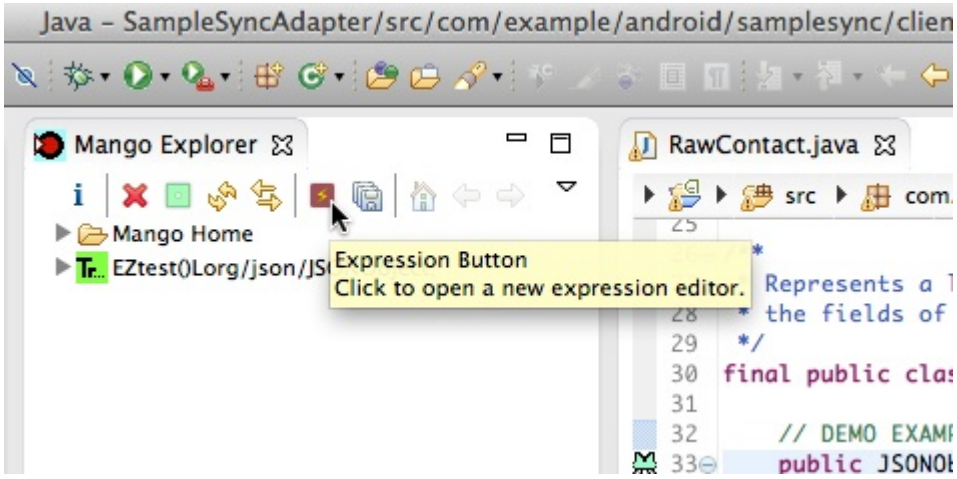
21 Open the "home state" expression. This may take about 30 seconds.



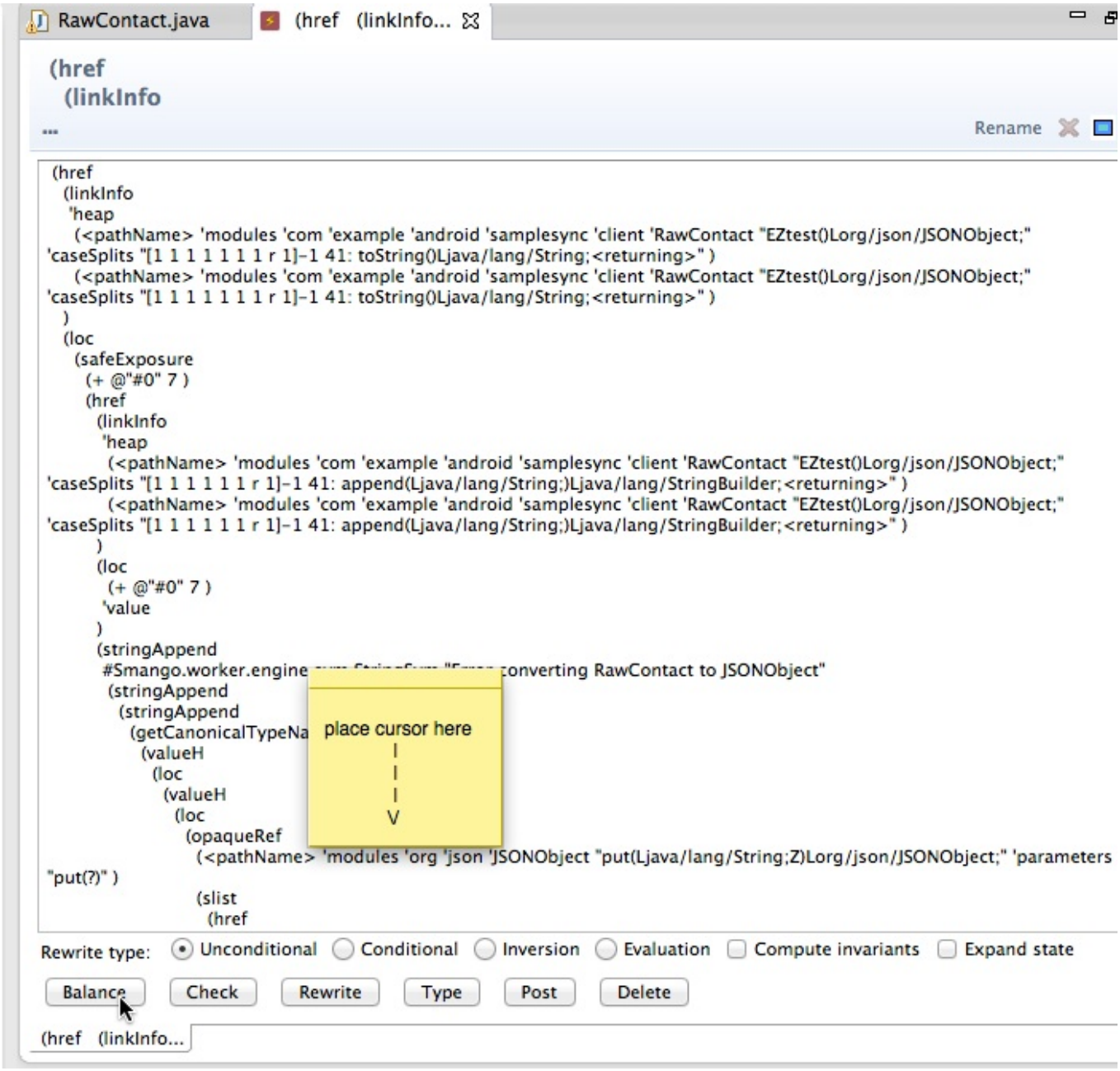
22 Open the mango console, scroll to the bottom and select the heap expression (see fig below).



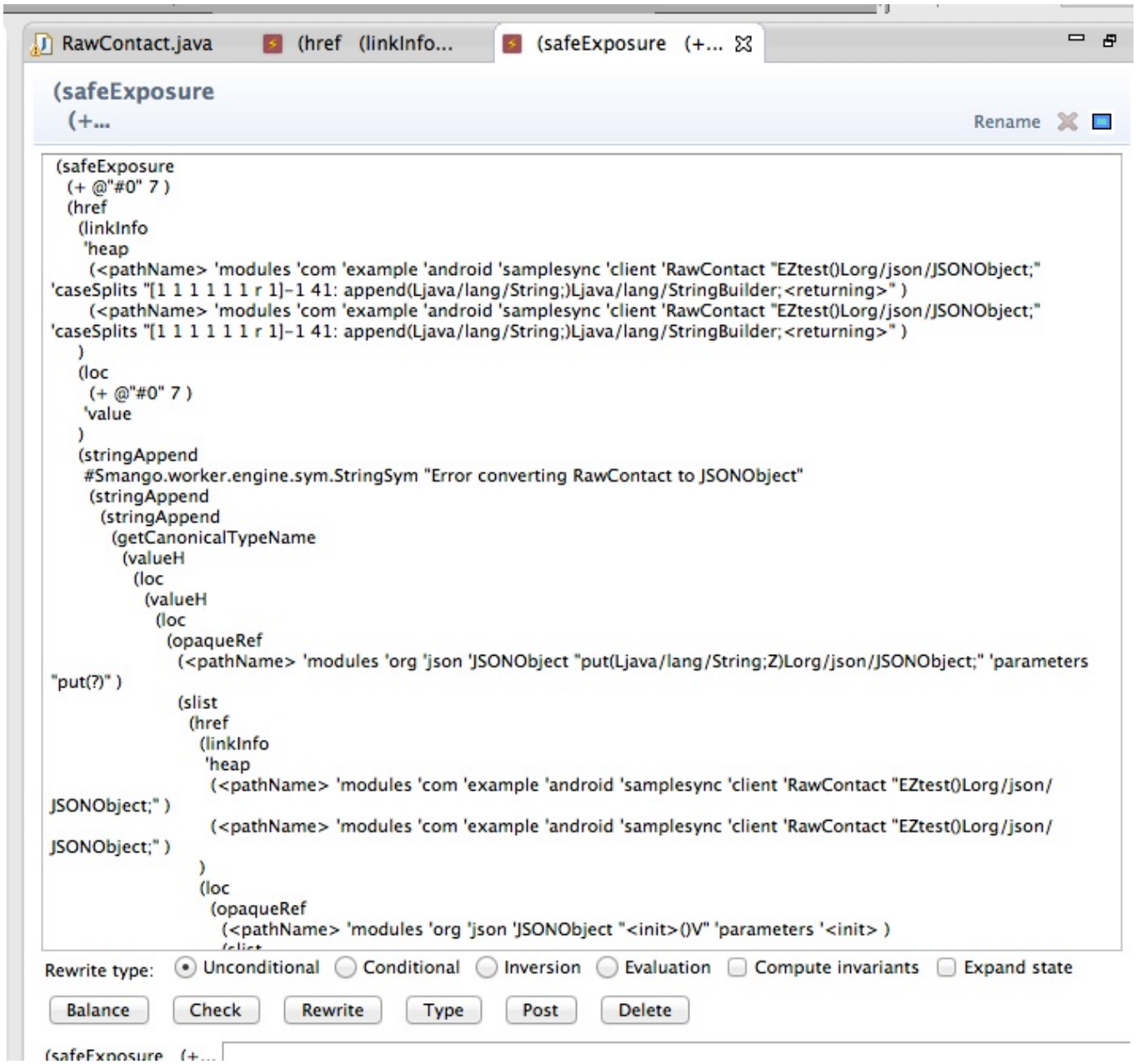
23 Copy the heap expression, and click on the Expression Editor button (see fig below).



24 The "Balance" command in the resulting Expression Editor captures a selection bounded the set of balanced parentheses closest to the cursor. Capture the (safeExposure …) expression by placing the text cursor as indicated and pushing the "Balance" button. Be sure the text cursor is on the same line as the "safeExposure" rule key, inside the open parenthesis. (see figure below)



25 Copy the selected safeExposure expression, and press the Expression Editor button. This will bring up the safeExposure expression inside its own editor window. (see figure below)



26 Generalize the safeExposure expression by selecting over-specified expressions with the "Balance" tool, and then typing in a distinct variable name which abstractly represents the expression. The name itself is immaterial, but it should be suggestive of the meaning of the abstracted expression, and different from other names introduced in this manner if and only if they refer to distinct sub-expressions. For example, the relevant fact about the put() method is that it is opaque, that is, its functionality is unknown. Therefore, replace the "pathName" of the put() method with "opaqueMethodName", see before and after figs below.

before

```
(safeExposure
  (+ @"#0" 7 )
  (href
    (linkInfo
     'heap
      (<pathName> 'modules 'com 'example 'android 'samplesync 'client 'RawContact "EZtest()Lorg/json/JSONObject;"
'caseSplits "[1 1 1 1 1 1 r 1]-1 41: append(Ljava/lang/String;)Ljava/lang/StringBuilder;<returning>" )
      (<pathName> 'modules 'com 'example 'android 'samplesync 'client 'RawContact "EZtest()Lorg/json/JSONObject;"
'caseSplits "[1 1 1 1 1 1 r 1]-1 41: append(Ljava/lang/String;)Ljava/lang/StringBuilder;<returning>" )
    )
    (loc
      (+ @"#0" 7 )
     'value
    )
    (stringAppend
     #Smango.worker.engine.sym.StringSym "Error converting RawContact to JSONObject"
      (stringAppend
        (stringAppend
          (getCanonicalTypeName
            (valueH
              (loc
                (valueH
                  (loc
                    (opaqueRef
                      (<pathName> 'modules 'org 'json 'JSONObject "put(Ljava/lang/String;Z)Lorg/json/JSONObject;" 'parameters
"put(?)" )
                      (slist
                        (href
                          (linkInfo
                           'heap
                            (<pathName> 'modules 'com 'example 'android 'samplesync 'client 'RawContact "EZtest()Lorg/json/
JSONObject;" )
                            (<pathName> 'modules 'com 'example 'android 'samplesync 'client 'RawContact "EZtest()Lorg/json/
```

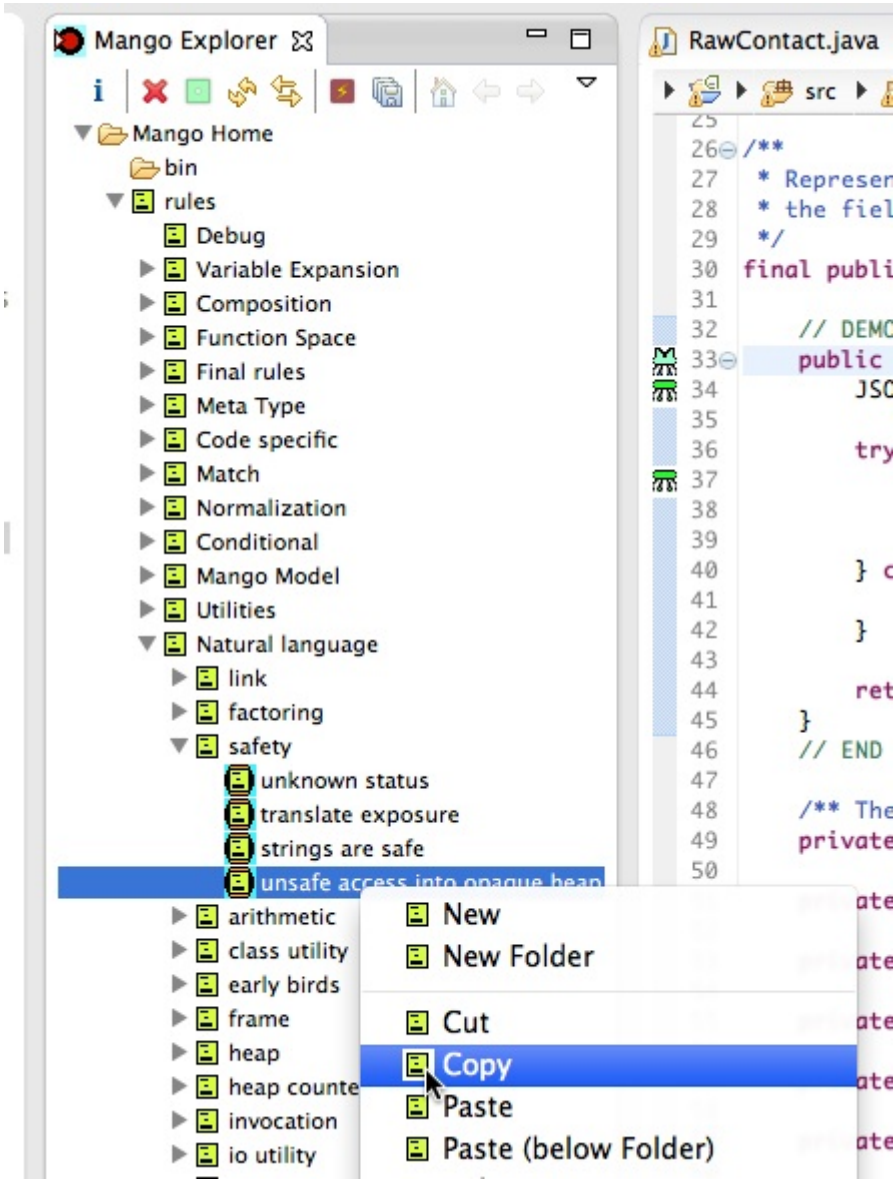
after

```
(safeExposure
  (+ @"#0" 7 )
  (href
    (linkInfo
     'heap
      (<pathName> 'modules 'com 'example 'android 'samplesync 'client 'RawContact "EZtest()Lorg/json/JSONObject;"
'caseSplits "[1 1 1 1 1 1 r 1]-1 41: append(Ljava/lang/String;)Ljava/lang/StringBuilder;<returning>" )
      (<pathName> 'modules 'com 'example 'android 'samplesync 'client 'RawContact "EZtest()Lorg/json/JSONObject;"
'caseSplits "[1 1 1 1 1 1 r 1]-1 41: append(Ljava/lang/String;)Ljava/lang/StringBuilder;<returning>" )
    )
    (loc
      (+ @"#0" 7 )
     'value
    )
    (stringAppend
     #Smango.worker.engine.sym.StringSym "Error converting RawContact to JSONObject"
      (stringAppend
        (stringAppend
          (getCanonicalTypeName
            (valueH
              (loc
                (valueH
                  (loc
                    (opaqueRef
                      opaqueMethodName
                      (slist
                        (href
                          (linkInfo
```

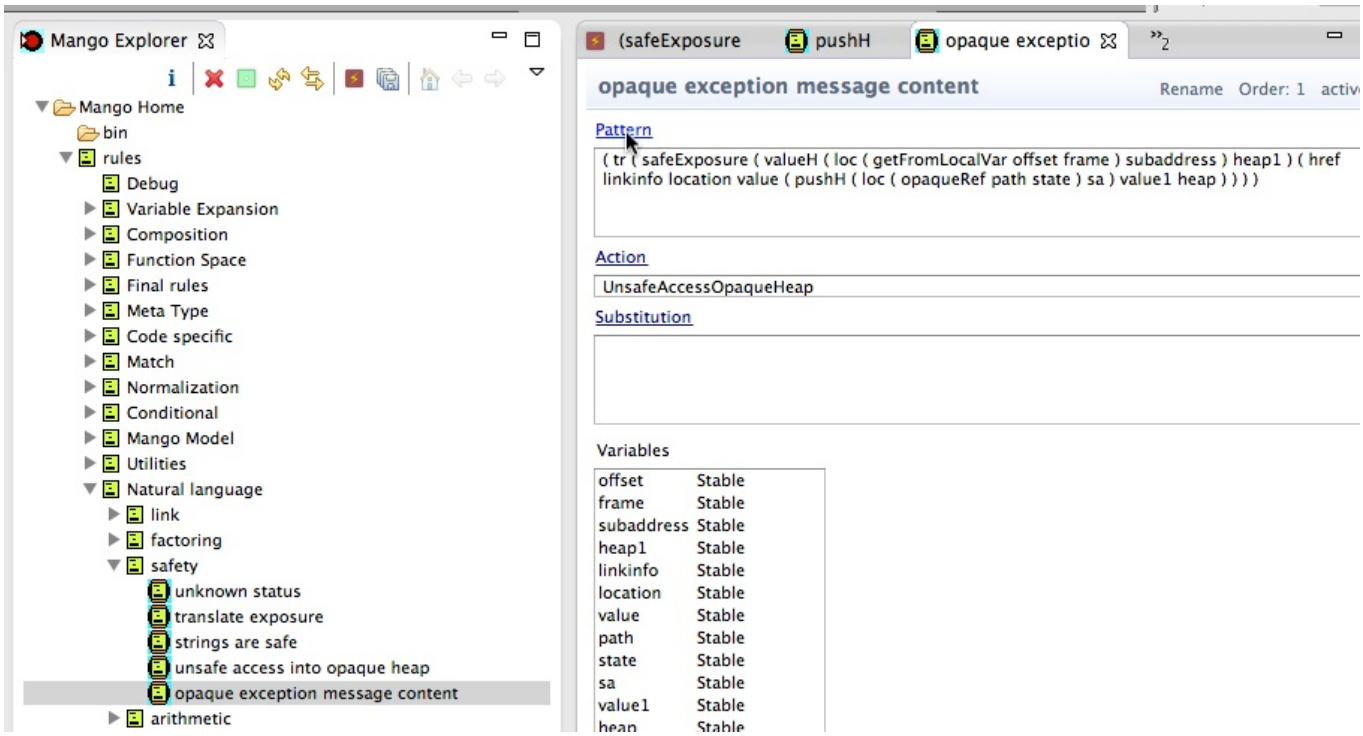
27 Continue in this manner until an expression similar to the one in the figure below is achieved. This process does involve a certain amount of guess-work. There are two possible errors. If the resulting expression is too general, then there will be false-positives, that is, expressions will be flagged as unsafe that are not really unsafe. On the other hand, if the resulting expression is too specific, then cases similar to this one will still be detected with "unknown exposure status". The suggested expression is about as general as possible with a pure substitution technique. A sharper result is possible but would require writing some code.

```
 (safeExposure
  ref
  (href
    linkinfo
    (loc
      ref
     'value
    )
    (stringAppend
     errorMsg
      (stringAppend
        (stringAppend
          (getCanonicalTypeName
            (valueH
              (loc
                (valueH
                  (loc
                    (opaqueRef
                     opaqueMethodName
                      methodInput
                    )
                   ^className
                  )
                  heap
                )
               ^className
              )
              heap2
            )
          )
         #Smango.worker.engine.sym.StringSym "@"
        )
        hexString
      )
    )
    heap3
  )
)
```

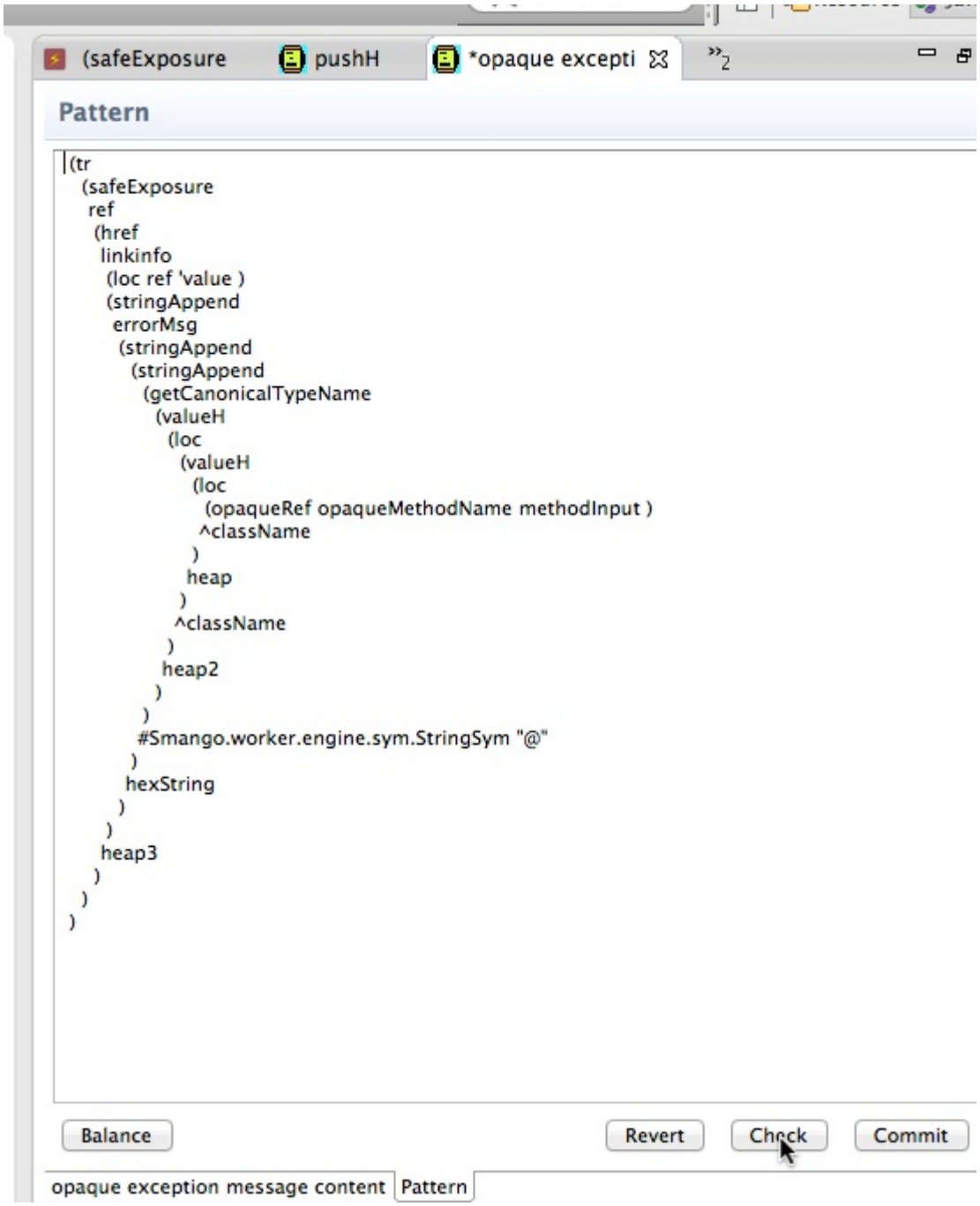
28 Navigate in the Mango Explorer to Mango Home/rules/Natural language/safety/unsafe access into opauque heap. Copy and past this rule, and rename the new rule to "opaque exception message content" (see figure below)



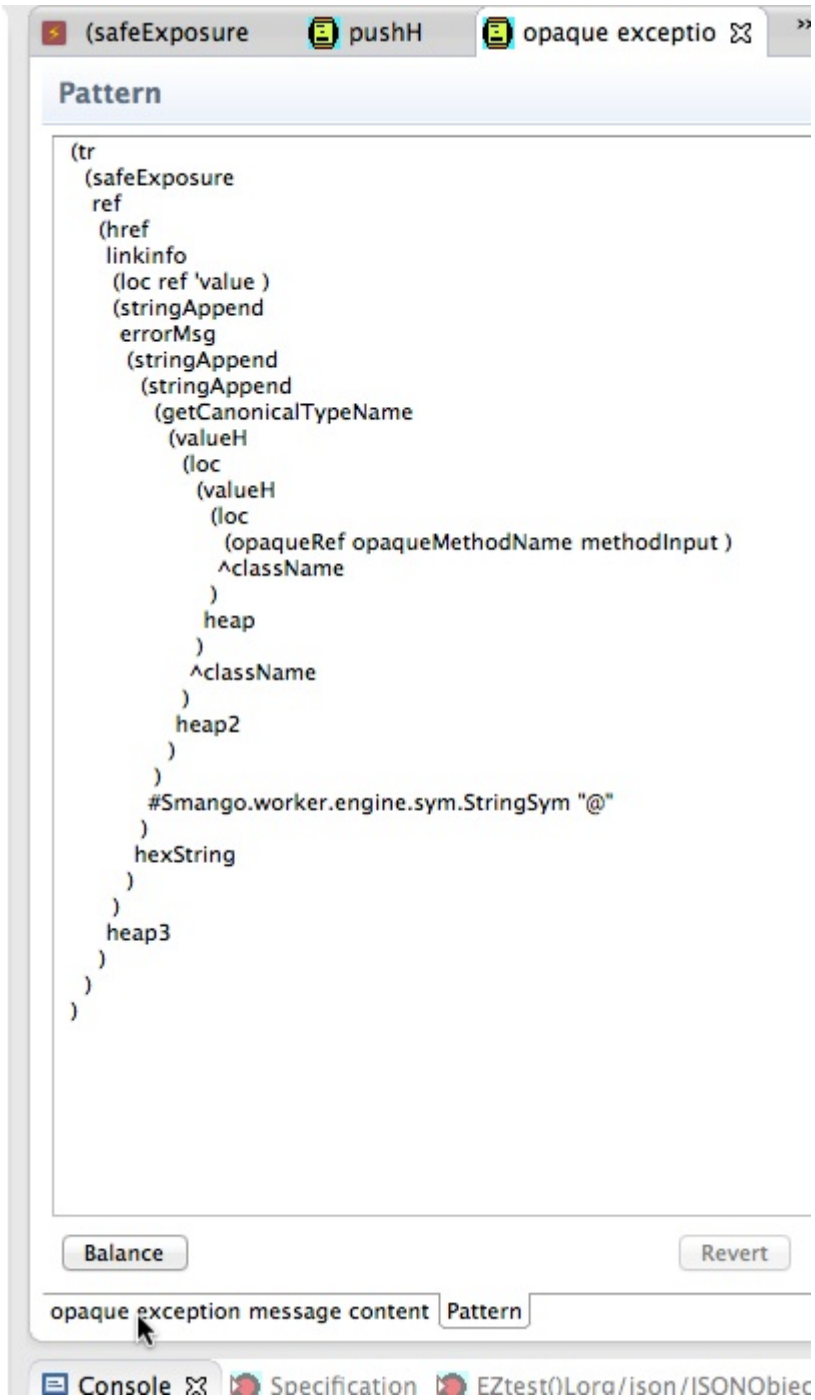
29 Open the new rule, and click on "pattern" (see figure below)



30 Replace the existing safeExposure expression with the newly created version. Be sure to leave the outer (tr ...) expression intact. Hit check to verify the syntax, and correct any mistake (see figure below)

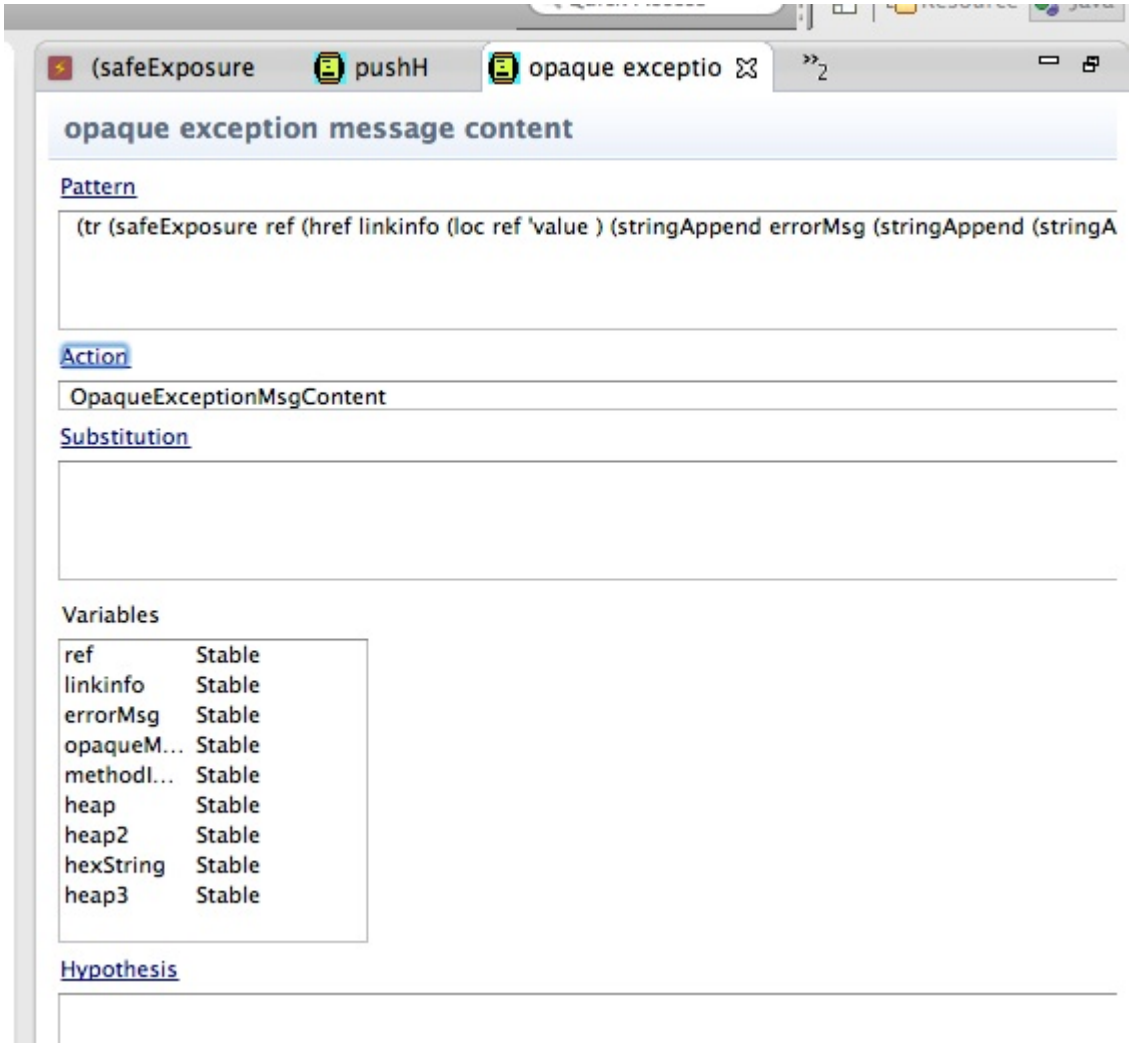


31 Save the rule, and return to the root tab by pressing the tab below the "Balance" button (see figure below)

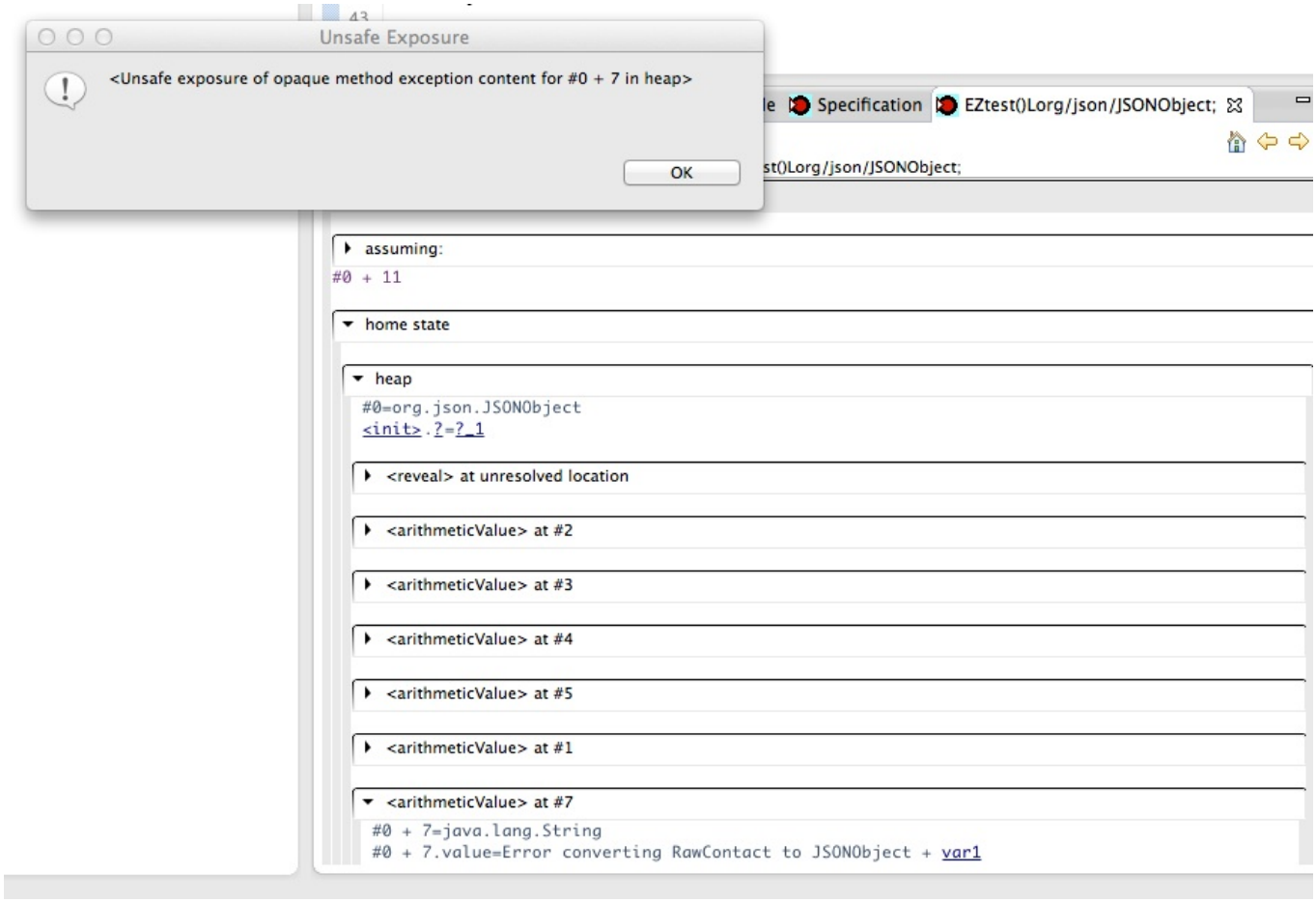


32 Locate the existing action code, "UnsafeAccessOpaqueHeap", in the developer platform. Copy and paste this class, and change the content as indicated (see figure below)

33 Returning to the Mango platform, click on the Action link in the "opaque exception message content" root tab, and navigate to the newly created action, in ruleAction/translate/safety. Push open, so that this new action now appears in the new rule. Save the rule. (see figure)



1. That's all there is to it! Close the specification view, close and reopen the Mango platform (to avoid a certain view persistence bug). Build the method population for EZ test, repeat steps 9-19, and observe the new rule will flag the unsafe exposure. (see figure) Notice the first few warnings don't have enough information to resolve to the new rule, but eventually it fires. Sweet success!


Unsafe Exposure
<Unsafe exposure of opaque method exception content for #0 + 7 in heap>
OK
Specification
EZtest()Lorg/json/JSONObject;
assuming:
#0 + 11
home state
heap
#0=org.json.JSONObject
<init>.?=?_1
<reveal> at unresolved location
<arithmeticValue> at #2
<arithmeticValue> at #3
<arithmeticValue> at #4
<arithmeticValue> at #5
<arithmeticValue> at #1
<arithmeticValue> at #7
#0 + 7=java.lang.String
#0 + 7.value=Error converting RawContact to JSONObject + var1