User Interface - Feature #1782

implement LAST-EVENT system handle and associated attributes

10/30/2012 09:32 AM - Greg Shah

Status:	Closed	**Start date:**	02/20/2013
Priority:	Normal	**Due date:**	10/25/2013
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	Runtime Support for Server Features		
billable:	No	**vendor_id:**	GCD

Description

Subtasks:

Feature # 2019: conversion support for LAST-EVENT system handle and attributes	**Closed**
Feature # 2020: runtime support for LAST-EVENT system handle and associated attributes	**Closed**

Related issues:

Related to User Interface - Feature #2226: finish LAST-EVENT support	**Closed**

History

#1 - 10/30/2012 09:32 AM - Greg Shah

This a handle that represents the most recent event that was received by the application. It has no methods, but it does have some attributes whose values vary by event type.

Please note that this may not be only a UI feature. Please test to see if non-UI events can be "seen" through this mechanism.

High priority attribute: LABEL

Low priority attributes:

CODE
COLUMN
EVENT-TYPE
FUNCTION
HANDLE
INSTANTIATING-PROCEDURE
ON-FRAME-BORDER
ROW
TYPE
WIDGET-ENTER
WIDGET-LEAVE
X
Y

#2 - 10/31/2012 02:56 PM - Greg Shah

- Target version set to Milestone 12

#3 - 02/20/2013 05:08 PM - Greg Shah

- Target version changed from Milestone 12 to Milestone 7

- Subject changed from implement LAST-EVENT to implement LAST-EVENT system handle and associated attributes

#4 - 02/23/2013 06:29 AM - Constantin Asofiei

- File ca_upd20130223b.zip added

LAST-EVENT handle is supported in P2J by using KeyReader.asHandle (when we have a h = LAST-EVENT case). For explicit LAST-EVENT attribute access, KeyReader API calls are emitted. There was a problem in conversion of LAST-EVENT:LABEL which is fixed in attached update.

#5 - 02/23/2013 10:26 AM - Constantin Asofiei

- File ca_upd20130223c.zip added

Merged with revision 10191.

#6 - 02/23/2013 10:29 AM - Greg Shah

Since the getLabel/setLabel is moved inside the check for LAST-EVENT, doesn't that disable using LABEL on a widget (or some other resource like buffer-fields that support it)?

#7 - 02/23/2013 10:41 AM - Constantin Asofiei

Correct, any attribute/method used with a widget should be in the main section too. I'm going through all the attrs/methods converted in the system handle sections and see which ones are shared with widgets too, to put them in the main section. The actual purpose of the sys handle specific sections is to emit static methods if the attr/method is used with the sys handle... for now I will copy-paste all widget-shared attr/methods from the sys handle sections to the main section, and put a TODO to get rid of the attr/method conversion from the sys handle specific sections (those should be used only to set a class qualifier in case the sys handle is used).

#8 - 02/23/2013 10:51 AM - Constantin Asofiei

- File ca_upd20130223d.zip added

Beside LABEL, the WIDTH/HEIGHT-CHARS/PIXELS attrs had to be duped in the main section too.

#9 - 02/23/2013 11:06 AM - Greg Shah

Looks good. It is going into conversion testing now.

#10 - 02/23/2013 12:06 PM - Greg Shah

This passed testing. Please check it in and distribute it.

#11 - 02/23/2013 12:13 PM - Constantin Asofiei

Committe to bzr revision 10192.

#12 - 10/11/2013 05:15 PM - Greg Shah

For now all we need to focus on is the LAST-EVENT:LABEL runtime support.

#13 - 10/11/2013 05:16 PM - Greg Shah

Constantin: what are your thoughts on what we need to investigate?

#14 - 10/12/2013 03:08 PM - Constantin Asofiei

Greg Shah wrote:

> Constantin: what are your thoughts on what we need to investigate?

We should check all kind of events which can be caught in a trigger:

1. widget events (leave, enter, etc)
2. key events
3. developer events (U1, U2, ..., U10 and CLOSE)
4. check how the LABEL is set after a UI statement (i.e. UPDATE/PROMPT-FOR/SET/MESSAGE ... UPDATE, etc)

Other cases:

1. events via APPLY statement (does it set the LAST-EVENT:LABEL to something?)
2. when is LABEL attribute set and reset (before entering the trigger? after? does it survive the trigger?)
3. all tests should be ran from the command line, to make sure the 4GL editor does not interfere in the behavior.
4. does PUBLISH/SUBSCRIBE events affect LAST-EVENT:LABEL?
5. double check if the LAST-EVENT:LABEL isn't somehow used in server-side events:

- Socket events: READ-RESPONSE, CONNECT. At least CONNECT event does set the label:

```
/* server program */
def var h as handle.
create server-socket h.

procedure proc1.
  def input param hh as handle.
  message LAST-EVENT:label. /* shows "CONNECT" string */
end.

h:enable-connections("-S 4433").

h:set-connect-procedure("proc1").

wait-for connect of h.

message last-event:label. /* shows "CONNECT" string */


/* client program */
def var h as handle.

create socket h.

h:connect("-S 4433").
```

- Async requests: PROCEDURE-COMPLETE event

#15 - 10/17/2013 05:26 PM - Vadim Gindin

I wrote several tests.
1. widget events and key events, apply statement

```
def var msg_watcher as char format "x(50)" no-undo.
def var fi as char format "x(50)" label "Fill-in" no-undo.
def var edit as char view-as editor size 10 by 2 no-undo.
def button exit auto-go label "Go to Exit".
def button test label "Test" size 15 by 3.
def button custbut label "call apply stmt".

def var txt0 as char format "x(75)" no-undo.
display "Feel free to move around..." view-as text
  skip fi skip (0.1)
  edit at 2 skip (0.1)
  test at 10
  exit at 50
  custbut at 90 skip (0.5)
  with frame f side-labels.

display txt0 label "last-event:label function (type) > self:type label"
with frame report 4 down centered title "Messages".

on tab, any-printable, go anywhere
  run msgwatch. /* This procedure uses last-event */

on entry, leave, go, return, choose of edit
  run msgwatch.

on U2 of fi
do:
  run msgwatch.
  display "U2" with frame report.
end.

/* RETURN CHOOSE [PROGRESS]-->FILL-IN Fill-in */
on choose of custbut
do:
  display "custbut" with frame report.
  run msgwatch.
  pause.
  apply "U2" to fi in frame f.
end.

enable all with frame f.

/* CHOOSE CHOOSE [PROGRESS]-->Fill-in field */
apply "U2" to fi in frame f.

wait-for go of frame f focus fi.

/* outputs LAST-EVENT info */
procedure msgwatch.
 txt0 = LAST-EVENT:LABEL + " " + LAST-EVENT:FUNCTION + " [" + LAST-EVENT:EVENT-TYPE + "]-->" + self:type + " "
.
 if can-query(self, "label")
 then txt0 = txt0 + self:label.
 else txt0 = txt0 + string(self).

 display txt0 with frame report.
 down with frame report.
end procedure.
```

It outputs fill-in field, edit field and 3 buttons. There are several triggers assigned on main events such as tab, return, entry, leave etc (developer event U2 is also used).
all events fill the LAST-EVENT:LABEL with correspondent event label.

Apply statement do not affects the behavior. The label remains the same (key pressed).

2. developer events. The same test case. The results are little different. label mostly the same except following moments:

a) EVENT-TYPE=PROGRESS for such events
b) for some cases the label is "choose". Probably it's appropriate event function in terms of documentation.

3. Here are couple of tests

```
def var x as int init 3.
def var b as char.
def var a as char.

b = LAST-EVENT:LABEL.
prompt-for x.
a = LAST-EVENT:LABEL.

message "Record buffer" x skip(0) "Screen buffer" input x.
message "b=" b "a= " a.
```

This test inits var x with number x (in a record buffer) and prompts the user to enter new value to write it to the a screen buffer. The old value in a record buffer remains unchanged.

```
for each customer:
  message "before: " LAST-EVENT:LABEL.
  update name.
  message "after: " LAST-EVENT:LABEL.
end.
```

Behavior is predictable in both cases. Labels is set dependent on the key pressed.

4. does PUBLISH/SUBSCRIBE events affect LAST-EVENT:LABEL?
no. These events don't set this var.

Open questions.
1. Am I on a write way? May be I should write some more specific tests?
2. "when is LABEL attribute set and reset (before entering the trigger? after? does it survive the trigger?)"
How to check it?
3. "double check if the LAST-EVENT:LABEL isn't somehow used in server-side events:"
What to do here?

#16 - 10/18/2013 06:09 PM - Vadim Gindin

need advice.

#17 - 10/19/2013 01:32 AM - Constantin Asofiei

Vadim Gindin wrote:

> I wrote several tests.
> 1. widget events and key events, apply statement
> ...
> Apply statement do not affects the behavior. The label remains the same (key pressed).

Good findings, a good place to start for the widget cases. But please test the ENDKEY/ENDERROR cases too.

> 2. developer events. The same test case. The results are little different. label mostly the same except following moments:
> a) EVENT-TYPE=PROGRESS for such events
> b) for some cases the label is "choose". Probably it's appropriate event function in terms of documentation.

Can this be because apply "U2" is ran from the choose trigger? Try to build a specific test for the developer events, so that the behavior is more predictable when reading the testcase (the previous test is pretty complex).

> 3. Here are couple of tests
> [...]
> This test inits var x with number x (in a record buffer) and prompts the user to enter new value to write it to the a screen buffer. The old value in a record buffer remains unchanged.

Just to note, for the PROMPT-FOR var case, a record buffer is not used; record buffer are used only when temp or physical records are referenced.

> Behavior is predictable in both cases. Labels is set dependent on the key pressed.

OK.

> Open questions.
> 1. Am I on a write way? May be I should write some more specific tests?

Yes, but try a little more experiments before starting implementing.

> 2. "when is LABEL attribute set and reset (before entering the trigger? after? does it survive the trigger?)"
> How to check it?

Here I was thinking to the APPLY statement in a trigger. Does the label before/after the APPLY is the same?

> 3. "double check if the LAST-EVENT:LABEL isn't somehow used in server-side events:"

For the socket events, you can use as starting point the example I gave. Also, the testcases/uast/sockets/socket-client.p and socket-server.p can be used as an example.
For the async request case, the test is a lot more complex (you need an appserver defined in 4GL and running it from P2J is not trivial). I'll check this and I'll post the results here.

#18 - 10/21/2013 04:08 PM - Vadim Gindin

ENDKEY,END-ERROR test:

```
on window-close of current-window stop.

for each customer on endkey undo, next: /* endkey, end-error - F4 */
  message "start - " LAST-EVENT:LABEL. /* outputs 'PF4' */
  display cust-num name credit-limit.
  set credit-limit validate(credit-limit > 0, "non-zero credit-limit").
  message "end - " LAST-EVENT:LABEL.
end.
```

Outputs credit limits for every customer from db and stops on each field letting the user change a value.
Outputs

```
start - RETURN
end - RETURN
```

If I press RETURN and outputs start - PF4 if I press F4.
Works as expected.

#19 - 10/21/2013 04:17 PM - Vadim Gindin

Developer events specific test.

```
def button custbut label "call apply stmt".
def var fi as char format "x(50)" label "Fill-in" no-undo.

def var txt0 as char format "x(75)" no-undo.

display "Feel free to move around..." view-as text
  fi skip
  custbut skip (0.5)
  with frame f side-labels.

display txt0 label "last-event:label function (type) > self:type label"
with frame report 4 down centered title "Messages".

on U3 of fi
do:
  run msgwatch.
  display "U3" with frame report.
end.
```

```
on U2 of fi
do:
  run msgwatch.
  display "U2" with frame report.
end.

/* RETURN CHOOSE [PROGRESS]-->FILL-IN Fill-in */
/*on choose of custbut
do:
  display "custbut" with frame report.
  run msgwatch.
  pause.
  apply "U2" to fi in frame f.
end.
*/

on leave of fi
do:
  apply "U3" to fi in frame f.
end.

enable all with frame f.

/* CHOOSE CHOOSE [PROGRESS]-->FILL-IN Fill-in */
/* apply "U2" to fi in frame f. */

wait-for go of frame f focus fi.

/* outputs LAST-EVENT info */
procedure msgwatch.
 if LAST-EVENT:LABEL <> ?
 then
  txt0 = LAST-EVENT:LABEL + " " + LAST-EVENT:FUNCTION + " [" + LAST-EVENT:EVENT-TYPE + "]-->" + self:type + "
".
 else
  txt0 = "empty last event label".
 if can-query(self, "label")
 then txt0 = txt0 + self:label.
 else txt0 = txt0 + string(self).

 display txt0 with frame report.
 down with frame report.
end procedure.
```

This test outputs the text field and button and applies "U2" event when the user presses the button and "U3" when the user leaves the text fields (see corresponding triggers). Here is the output:

```
TAB LEAVE [KEYPRESS] --> FILL-IN Fill-in                          U3
RETURN CHOOSE [PROGRESS] --> FILL-IN Fill-in                      U2
```

So you were right. But at this moment I don't understand why the "choose" event is so specific..

#20 - 10/21/2013 04:24 PM - Constantin Asofiei

Vadim Gindin wrote:

> So you were right. But at this moment I don't understand why the "choose" event is so specific..

When dealing with events, running the program from the editor may lead you to wrong paths. Please run all tests from the command line, using i.e. pro -p leave-test.p.
In your case, I think the CHOOSE was because of the editor; this does not appear after running the program from command line.

#21 - 10/22/2013 08:45 AM - Vadim Gindin

I ran scripts exactly in this way as Constantin wrote earlier.

#22 - 10/22/2013 03:22 PM - Vadim Gindin

another portion of tests
1. before trigger, inside trigger and after trigger.

```
def button custbut label "some button".
def button superbut label "try developer event".

def var fi as char format "x(50)" label "Fill-in" no-undo.

display "Feel free to move around..." view-as text
  fi skip
  custbut skip
  superbut skip
  with frame f side-labels.

on choose of custbut
do:
  message 'inside choose: ' LAST-EVENT:LABEL.
end.

on leave of fi
do:
  message 'before choose: ' LAST-EVENT:LABEL.
  apply "choose" to custbut in frame f.
  message 'after choose: ' LAST-EVENT:LABEL.
end.

on choose of superbut
do:
  message 'before U4: ' LAST-EVENT:LABEL.
  apply "U4" to superbut in frame f.
  message 'after U4: ' LAST-EVENT:LABEL.
end.

on U4 of superbut
do:
  message 'inside U4: ' . LAST-EVENT:LABEL.
end.

enable all with frame f.

wait-for go of frame f focus fi.
output close.
```

UI consists of text field and two buttons. Simple widget events is triggered in a text field "leave" trigger and developer event is triggered on a button choose event.

Result for widget events was as expected: before, inside, after - in all cases the value is the same (TAB). But result for the developer event U4 is little different: before=after=RETURN but inside is empty. The label inside developer event is empty.

2. socket events are working as expected: outputs CONNECT two times.

#23 - 10/22/2013 03:51 PM - Vadim Gindin

If I correctly understand the last case is "Async requests: PROCEDURE-COMPLETE event". How to test it without AppServer? Here are the "RUN" doc:

```
RUN
   {  extern-proc-name
   ...
   }
   ...
   [ON [SERVER] { server-handle | session-handle }
       [ASYNCHRONOUS
           ...
       ]
   ]
```

It means that I can run asynchronous procedure only on the server or separate session. Could you advice me the appropriate way?

#24 - 10/23/2013 04:14 AM - Constantin Asofiei

Vadim Gindin wrote:

> another portion of tests
> 1. before trigger, inside trigger and after trigger.
> [...]
>
> UI consists of text field and two buttons. Simple widget events is triggered in a text field "leave" trigger and developer event is triggered on a button choose event.
>
> Result for widget events was as expected: before, inside, after - in all cases the value is the same (TAB). But result for the developer event U4 is little different: before=after=RETURN but inside is empty. The label inside developer event is empty.

Something else to test inside a trigger: do another readkey or UI statement, which can change the label. At this time, to me it looks like there is a backup/restore mechanism for this label (i.e. label is saved before a UI/trigger is invoked and restored afterwords).

More, check the EDITING phrase too - does a READKEY set the LABEL?

2. socket events are working as expected: outputs CONNECT two times.

Please test the READ-RESPONSE event to.

#25 - 10/23/2013 04:51 AM - Constantin Asofiei

Vadim Gindin wrote:

> If I correctly understand the last case is "Async requests: PROCEDURE-COMPLETE event". How to test it without AppServer? Here are the "RUN" doc:
> [...]
>
> It means that I can run asynchronous procedure only on the server or separate session. Could you advice me the appropriate way?

We have two cases for the async requests: running a proc on SESSION or on a remote appserver. Run them from command line:

1. running on SESSION:

```
def var ha as handle.

procedure proc1.
   message "bla" last-event:label.
end.

run test.p on server session asynchronous set ha event-procedure "proc1".

pause 1.

wait-for procedure-complete of ha.
```

   In this case, I think the LAST-EVENT:LABEL is not set by the async call. Make sure you have an empty test.p file. Please experiment a little more to make sure I got it right.
2. running on an appserver (test on WINDEV01 please):

```
def var hs as handle.
def var hp as handle.
def var ha as handle.
def var ch as char.

procedure proc1.
   message last-event:label.
end.

create server hs.
message "connected:" hs:connect("-S 2223 -DirectConnect -H localhost -sessionModel session-free").

run test.p persistent set hp on server hs asynchronous set ha event-procedure "proc1".
run proc0 in hp asynchronous set ha event-procedure "proc1".

wait-for procedure-complete of ha.

message "disconnected:" hs:disconnect().
```

   In both calls, the label looks like is set to PROCEDURE-COMPLETE.

#26 - 10/25/2013 06:18 AM - Vadim Gindin

Constantin Asofiei wrote:

> Something else to test inside a trigger: do another readkey or UI statement, which can change the label. At this time, to me it looks like there is a backup/restore mechanism for this label (i.e. label is saved before a UI/trigger is invoked and restored afterwords).

Actually I don't understand what the phrase "do another readkey" means. What is readkey here? I tried to change widget events but the result is the same.

> More, check the EDITING phrase too - does a READKEY set the LABEL?

I wrote simple test:

```
def button rd label "red".

display rd with frame f side-labels.

on choose of rd
do:
    message "before: " LAST-EVENT:LABEL.
    readkey pause 5.
    message "after: " LAST-EVENT:LABEL.

end.

enable all with frame f.
wait-for go of frame f.
```

The result is as predictable:

```
before: RETURN
after: RETURN
```

> 2. socket events are working as expected: outputs CONNECT two times.

> Please test the READ-RESPONSE event to.

I tried to add an READ-RESPONSE event processing in existing test for the CONNECT label and I got an error: "SET-READ-RESPONSE-PROCEDURE is not queryable attribute for SERVER-SOCKET widget. (4052)". When I tried to add this call to the client procedure - corresponding procedure didn't called and returned nothing.

I also tested async requests and got that the first test (SESSION case) don't outputs a message but the second case (with AppServ) really outputs "PROCEDURE-COMPLETE".

#27 - 10/25/2013 09:09 AM - Constantin Asofiei

> Actually I don't understand what the phrase "do another readkey" means. What is readkey here? I tried to change widget events but the result is the same.

I'm mean we have to test the READKEY too. In your case, change it like this:

```
def button rd label "red".

display rd with frame f side-labels.

on choose of rd
do:
    message "before: " LAST-EVENT:LABEL.
    readkey. /* press any key */
    message "after: " LAST-EVENT:LABEL.

end.

enable all with frame f.
wait-for go of frame f.
```

Also, in a trigger we can have a nested UI statement:

```
def button rd label "red".
def var i as int.

form i with frame f1.

display rd with frame f side-labels.

on 1 anywhere do:
    message "triger '1': " LAST-EVENT:LABEL.
end.

on choose of rd
do:
    message "before: " LAST-EVENT:LABEL.
    update i with frame f1. /* prese 1 here */
    message "after: " LAST-EVENT:LABEL.
end.

enable all with frame f.
wait-for go of frame f.
```

For this case, AFTER shows the same label as BEFORE; that's why I mentioned I have a feeling the label has a backup/restore mechanism.

> More, check the EDITING phrase too - does a READKEY set the LABEL?

The EDITING phrase looks like this:

```
def var i as int.

update i with frame f1 editing:
  message "1:" last-event:label.
  readkey.
  message "2:" last-event:label.
  apply last-key.
  message "3:" last-event:label.
end.
```

BTW, have you tested MESSAGE ... UPDATE and MESSAGE ... VIEW-AS ALERT-BOX cases?

> I tried to add an READ-RESPONSE event processing in existing test for the CONNECT label and I got an error: "SET-READ-RESPONSE-PROCEDURE is not queryable attribute for SERVER-SOCKET widget. (4052)". When I tried to add this call to the client procedure - corresponding procedure didn't called and returned nothing.

For READ-RESPONSE, see the testcases/uast/sockets/socket-server.p and socket-client.p files. In socket-client.p there is a "proc0" procedure, where you can check the label.

#28 - 10/29/2013 02:33 PM - Vadim Gindin

Here are results of remained testing:

1. Readkey. The result of your test and mine (with pause) shows that there is no restore/backup mechanism or something like that. before and after shows the key, that user pressed:

```
before: RETURN // because of button press
after: f // user pressed 'f'
```

or

```
before:   // because the user pressed the space.
after: f // the user pressed 'f'
```

2. nested UI statements. The result is also predictable. LAST-EVENT:LABEL contains the last key pressed.

```
before: RETURN // because of the button press
trigger '1': 1 // because of the user pressed '1' and our trigger fired.
after: RETURN // // because the user pressed RETURN
```

I suppose that backup/restore mechanism isn't used here. The case is that UPDATE statement do a lot of work and at the end it blocks the application while waiting for GO (RETURN). So we can go out from the field only pressing RETURN or ENDKEY (F4). In our case if the user press F4 the field will loose the focus and the button will get it. Field changes will be discarded. In our test nothing will be displayed. But lets add additional var j to the test:

```
def button rd label "red".
def var i as int.
def var j as int.

form i skip j with frame f1.

display rd with frame f side-labels.

on 1 anywhere do:
    message "triger '1': " LAST-EVENT:LABEL.
end.

on choose of rd
do:
    message "before: " LAST-EVENT:LABEL.
    update i with frame f1. /* prese 1 here */
```

```
    update j with frame f1.
    message "after: " LAST-EVENT:LABEL.
end.

enable all with frame f.
wait-for go of frame f.
```

Here we'll see the difference. First we press RETURN and went to the field i. OUTPUT will contain before: RETURN. We are writing some value and pressing RETURN. As the result we are going to the field 2 (for the var j). The output remains the same. We also writing some value and pressing ENDKEY (F4) and that's it! The output will contain: before: PF4

So in this case the update statement are limiting available test cases but the Progress behaves predictable and LAST-EVENT:LABEL shows the last key function.

3. EDITING phrase. I'm recalling that this phrase is intended to process field editing by every char typed by the user.

At the first start:

```
1:
2: 8
3: 8
```

or following:

```
1: 7
2: 7
3: 7
```

I.e. READKEY don't set the LAST-EVENT:LABEL. It just don't "touch" it.

4. MESSAGE .. UPDATE and MESSAGE .. VIEW-AS ALERT-BOX statements.

MESSAGE UPDATE test:

```
def button rd label "red".
def var i as int init 67.

display rd with frame f side-labels.

on choose of rd
do:
    message i " before : " LAST-EVENT:LABEL update i.
    readkey pause 2.
    message i "after: " LAST-EVENT:LABEL update i.

end.

enable all with frame f.
wait-for go of frame f.
```

The result is interesting. Starting:
0 before: RETURN +0 +.
Then write 45 and press RETURN. The result will be:
0 after: CTRL-SHIFT-ESC-HELP-KEY +0 +

I don't know how to interpret this result. Probably here is an incorrect combination of statements. Because if we remove readkey from this test case - all will work.

Here is the test case with MESSAGE .. VIEW-AS ALERT-BOX:

```
def button rd label "red".
```

```
def var i as int init 67.

display rd with frame f side-labels.

on choose of rd
do:
    message " before : " LAST-EVENT:LABEL view-as alert-box.
    readkey pause 5.
    message "after: " LAST-EVENT:LABEL view-as alert-box.

end.

enable all with frame f.
wait-for go of frame f.
```

This test works as expected with readkey and without it. Alert boxes show the actual last key pressed by the user.

5. The final tests are about socket events (CONNECT and READ-RESPONSE). I modified the test cases you specified a little:
server-socket.p:

```
OUTPUT TO server-socket-out.

DEFINE VARIABLE serverSocket AS HANDLE.
DEFINE VARIABLE aOk AS LOGICAL.
DEFINE VARIABLE serverWriteBuffer AS MEMPTR.
DEFINE VARIABLE serverReadBuffer AS MEMPTR.
DEFINE VARIABLE serverMessage AS CHAR.
MESSAGE "We are in the socket server".

CREATE SERVER-SOCKET serverSocket.
/* Marshal data to write */
SET-SIZE(serverWriteBuffer) = 64.
SET-SIZE(serverReadBuffer) = 64.
serverMessage = "SERVER - hello".
PUT-STRING(serverWriteBuffer,1) = serverMessage.
MESSAGE "Start event handling".

serverSocket:SET-CONNECT-PROCEDURE( "connProc").
aOk = serverSocket:ENABLE-CONNECTIONS( "-S 3363").
MESSAGE "Enabled connections:" aOk.
IF NOT aOk THEN
RETURN.

/* This WAIT-FOR completes after the first connection */
WAIT-FOR CONNECT OF serverSocket.
MESSAGE "FIRST CONNECTION RECEIVED".

serverSocket:DISABLE-CONNECTIONS().
DELETE OBJECT serverSocket.
MESSAGE "Finished".

/* Connection procedure for server socket */
PROCEDURE connProc.
   /*Socket recieved as parameter*/
   DEFINE INPUT PARAMETER hSocket AS HANDLE.
   MESSAGE "We are in CONNECT event procedure, connProc".
   hSocket:WRITE (serverWriteBuffer, 1, LENGTH(serverMessage)).
   MESSAGE hSocket:BYTES-WRITTEN "bytes written".
   /* Wait for response from client */
   MESSAGE "server before: " LAST-EVENT:LABEL.
   WAIT-FOR READ-RESPONSE OF hSocket.
   MESSAGE "server after: " LAST-EVENT:LABEL.
   hsocket:READ(serverReadBuffer,1,hsocket:GET-BYTES-AVAILABLE()).
   DEFINE VARIABLE clientMessage AS CHAR.
   clientMessage = GET-STRING(serverReadBuffer,1).
   DISPLAY clientMessage FORMAT "x(64)".
END.
```

and client-socket.p

```
OUTPUT TO client-socket-out.

DEFINE VARIABLE clientSocket AS HANDLE.
DEFINE VARIABLE acknowledge AS LOGICAL.
DEFINE VARIABLE clientReadBuffer AS MEMPTR.
DEFINE VARIABLE clientWriteBuffer AS MEMPTR.
DEFINE VARIABLE cString AS CHARACTER.
DEFINE VARIABLE clientMessage AS CHARACTER.
MESSAGE "We are in socket client".

CREATE SOCKET clientSocket.
clientSocket:CONNECT ("-H localhost -S 3363").
IF clientSocket:CONNECTED() THEN
   MESSAGE "Connected OK".
ELSE DO:
   MESSAGE "Could not connect".
   RETURN.
END.

/* Do READ-RESPONSE event handling */
MESSAGE "Start event handling".
clientSocket:SET-READ-RESPONSE-PROCEDURE( "readProc").
/* This WAIT-FOR completes after the first data reception */
WAIT-FOR READ-RESPONSE OF clientSocket.

MESSAGE "Freeing up resources".
clientSocket:DISCONNECT().
DELETE OBJECT clientSocket.
MESSAGE "Finished".

/* Read procedure for socket */
PROCEDURE readProc.
   DEFINE VARIABLE clientReadBuffer AS MEMPTR.
   MESSAGE "We are in READ-RESPONSE event procedure, readProc".
   SET-SIZE(clientReadBuffer) = 64.
   SET-SIZE(clientWriteBuffer) = 64.
   clientSocket:READ(clientReadBuffer,1,clientSocket:GET-BYTES-AVAILABLE()).

   PAUSE 5.
   MESSAGE "client. after read: " LAST-EVENT:LABEL.

   /*In the socket procedure SELF will point to the handle of the socket for
   which this procedure is set */
   MESSAGE clientSocket:BYTES-READ "bytes read".
   /*Unmarshal data*/
   cString = GET-STRING(clientReadBuffer,1).
   DISPLAY cString FORMAT "x(64)".
   /* send data to server*/
   MESSAGE "send back data to the server".
   clientMessage = "Client - hi".
   PUT-STRING(clientWriteBuffer,1) = clientMessage.

   PAUSE 5.
   MESSAGE "client. before write: " LAST-EVENT:LABEL.

   clientSocket:WRITE (clientWriteBuffer,1,LENGTH(clientMessage)).
END PROCEDURE.
```

I ran them on windev01 and got the following results in output files:
server-socket-out:

```
We are in the socket server
Start event handling
Enabled connections: yes
We are in CONNECT event procedure, connProc
14 bytes written
server before:  CONNECT
server after:  READ-RESPONSE
```

```
clientMessage-------------------------------------------------------

Client - hi
FIRST CONNECTION RECEIVED
Finished
```

and

client-socket-out:

```
We are in socket client
Connected OK
Start event handling
We are in READ-RESPONSE event procedure, readProc
client. after read:  CTRL-SHIFT-ALT-HELP-KEY
14 bytes read

cString-------------------------------------------------------------

SERVER - hello
send back data to the server
client. before write:  CTRL-SHIFT-ALT-HELP-KEY
Freeing up resources
Finished
```

The client and the server are exchanging the messages and outputs debug information in message area. My modifications are only additional message statements with LAST-EVENT:LABEL.

I paste it in the server procedure in the connection sub-procedure "conProc" before and after WAIT-FOR statement for read response. And as the result the server output LABEL contains before: CONNECT and after: READ-RESPONSE.

I also paste it in the client procedure in the read-response sub-procedure "readProc". And the client output LABEL contains: client. after read: CTRL-SHIFT-ALT-HELP-KEY and client. before write: CTRL-SHIFT-ALT-HELP-KEY

I don't know.. probably this value for the LAST-EVENT:LABEL is reserved by Progress and have some special meaning.

How do you think is this test cases set enough?

1. Readkey

OK

2. nested UI statements

I think you are right, and the following shows this without using ENDKEY:

```
on 3 anywhere do:
   apply "go" to frame f1.
end.
on choose of rd
do:
    message "before: " LAST-EVENT:LABEL. /* this is "RETURN" */
    update i with frame f1. /* press 1 here then return */
    update j with frame f1. /* press 2 here then 3 */
    message "after: " LAST-EVENT:LABEL. /* this is "3" */
end.
```

Also, for the ENDKEY (F4) case, tested like this can show the "after" label if the END-ERROR condition is caught:

```
on choose of rd
do:
    message "before: " LAST-EVENT:LABEL. /* this is "RETURN" */
    do on error undo, leave: /* this will catch the ENDKEY */
       update i with frame f1. /* press 1 here then return */
       update j with frame f1. /* press 2 here then ENDKEY/F4 */
    end.
    message "after: " LAST-EVENT:LABEL. /* this is "PF4" */
end.
```

This is good news, as it shows that LAST-EVENT:LABEL is forward-only.

3. EDITING phrase

OK

4. MESSAGE .. UPDATE and MESSAGE .. VIEW-AS ALERT-BOX statements.

The key here is the READKEY PAUSE statement. Using this simple test:

```
readkey pause 1.
message last-event:label.
```

this will be shown:

```
CTRL-SHIFT-ESC-HELP-KEY
```

I guess this might be because when the pause ends, it triggers some kind of event... because we have the same output for this code:

```
pause 1.
message last-event:label.
```

Anyway, this is an interesting finding, but is low priority.

For the MESSAGE ... VIEW-AS ALERT-BOX, there is the BUTTONS clause too:

```
message "Something something" view-as alert-box buttons yes-no-cancel.
message last-event:label.
```

If shortcuts are used (press y, n or c), you will see that "PF1" is displayed in both cases (so this is good to now, because the label isn't actually what the user pressed).

When testing a statement, I think is better to start with simple tests, like:

```
message "enter something:" update i as int.
message last-event:label.
```

and after that move on to complex tests; by observing simple behavior first and complex behavior second, I think is easier to see the "big picture".

Also, something else about tests: lately (and thanks to Evgeny K., I got the idea after looking through one of his tests), I'm writing the tests like:

```
message "Something something" view-as alert-box buttons yes-no-cancel.
if last-event:label <> "PF1" then message "problem in LAST-EVENT:LABEL for MESSAGE ... VIEW-AS ALERT-BOX".
```

Idea is, if possible, do not rely on the side-by-side comparing of a P2J and a 4GL terminal running the same case. This way, the tests will act as "unit tests", output will be done only in case of errors and regressions are a lot easier to find.

5. The final tests are about socket events

The socket testing is OK, but please check the LAST-EVENT:LABEL at the begining of the read-response/connect procedure too.

> I also paste it in the client procedure in the read-response sub-procedure "readProc". And the client output LABEL contains: client. after read: CTRL-SHIFT-ALT-HELP-KEY and client. before write: CTRL-SHIFT-ALT-HELP-KEY

I'm almost sure this has something to do with the PAUSE statement.

Also, while running your socket tests I found something else: when the message area needs to be cleared and a key needs to be pressed, that key will be recorded as the LAST-EVENT:LABEL:

```
def var i as int.
message "read a key:".
readkey.

do i = 1 to 5:
 message string(i) last-event:label.
end.
```

> How do you think is this test cases set enough?

Yes, please start implementing them in this order:

1. UI statements and triggers (nested or not)
2. EDITING phrase
3. READKEY (no pause)
4. MESSAGE, MESSAGE UPDATE, MESSAGE VIEW-AS ALERTBOX, pause between message statements
5. developer events

6. PAUSE, READKEY PAUSE
7. READ-RESPONSE, CONNECT and PROCEDURE-COMPLETE

If I've missed something let me know. Also, do post updates for each sub-issue, so we can review it.

#30 - 10/30/2013 09:57 AM - Greg Shah

The CTRL-SHIFT-ALT-HELP-KEY may be a kind of "ANY" key that is reported from PAUSE.

#31 - 11/01/2013 02:22 PM - Vadim Gindin

Here is a principled problem. The documentation says that the value of LAST-EVENT:LABEL depends on LAST-EVENT:EVENT-TYPE:

```
For the LAST-EVENT handle, the LABEL attribute returns the names of low-level events based on the EVENT-TYPE a
ttribute value. For EVENT-TYPE="KEYPRESS", this attribute returns key label events, such as "F1" or "ESC". It
also returns key labels of any keys that trigger key function events (returned by the FUNCTION attribute).

For EVENT-TYPE="MOUSE", this attribute returns low-level events for both portable and three-button mouse event
 types, such as "SELECT-MOUSE-UP" (portable) or "LEFT-MOUSE-UP" (three-button). It also returns the names of t
he low-level mouse actions that trigger any high-level mouse-events (returned by the FUNCTION attribute).

For EVENT-TYPE="PROGRESS", this attribute returns the same high-level event name returned by the FUNCTION attr
ibute unless the Progress event is triggered by a key press. In this case it returns the key label of the key
that triggered the event.

NOTE: When the AUTO-RESIZE attribute is set to TRUE, Progress resized button and toggle-box widgets with run-t
ime changes to the LABEL attribute.
```

But EVENT-TYPE is not implemented yet..

The first impl variant I can propose is this:

```
KeyReader:
   public static character getLabel()
   {
      return new character(Keyboard.keyLabel(workArea.obtain().lastkey));
   }
```

It should work for EVENT-TYPE="KEYPRESS" variant.. and not only for the UI widget events but for others. What do you think.

#32 - 11/03/2013 09:20 AM - Greg Shah

> But EVENT-TYPE is not implemented yet..

Please enhance your testcases to add output of EVENT-TYPE and FUNCTION (where applicable). Document the results here. That will help us understand how accurate the Progress docs are and what the implications of EVENT-TYPE will be for LAST-EVENT.

> It should work for EVENT-TYPE="KEYPRESS" variant.. and not only for the UI widget events but for others. What do you think.

I'm generally OK with this, but I would want to make sure it provides fully compatible results for all your testcases so far. I'm not sure that is the case.

#33 - 11/04/2013 04:10 PM - Vadim Gindin

1. "anywhere" is not supported
2. widgets events works fine with current impl of LAST-EVENT:LABEL
3. EVENT-TYPE is not supported by grammar yet (testcases are not converted)

#34 - 11/06/2013 12:15 PM - Vadim Gindin

Here are some test results (I added EVENT-TYPE).

EVENT-TYPES
1. editing phrase - KEYPRESS
2. message .. view-as alert-box
before: RETURN [PROGRESS] // button press (FUNCTION:CHOOSE)
after: RETURN [PROGRESS]
3. message .. update
before: RETURN [PROGRESS] // button press (FUNCTION:CHOOSE)
after: RETURN [KEYPRESS]
5. nested
before: RETURN [PROGRESS] // button press (FUNCTION:CHOOSE)
after: RETURN [KEYPRESS]
1: RETURN [KEYPRESS]
6.lev_bef_aft (widget events end dev events) before and after
same
7. readkey
same
8. endkey
PF4 [KEYPRESS] (where PF4 is a LABEL)
9. lev_dev_ev.p (developer events)
same
10. lev_update_ev.p (update statement)
same
11. update_ev.p (prompt-for stmt)
same
12. sess_proc (asynchronous call)
LABEL is empty and EVENT-TYPE is empty too.

As a result we have that the common case is EVENT-TYPE=KEYPRESS and only "button press" event has EVENT-TYPE=PROGRESS in these

tests. I couldn't check server-client tests (on windev01) because of connection problem, but it seems that it's a common case.

By the way, converted test "msgalert" is working different than corresponding Progress procedure. Converted Java application outputs "after: ". I.e. LAST-EVENT:LABEL is empty in this case after message .. view-as alert-box statement.

#35 - 11/08/2013 05:15 PM - Vadim Gindin

The more I'm executing tests (source and converted) the more different I'm getting results.
The last actual result is the proposed implementation matches 99% to the source behavior. 1% of differences is following:
1) The test "editing phrase" shows empty string as LAST-EVENT:LABEL (Progress) and PF1 (Java).
2) The test "editing phrase". Press 4, than again 4, than F4. Progress shows "PF4" but Java shows "F4".
3) How to test converted procedures that requires two parts: client and server, sessProc and test? I couldn't test such cases.

What to do next?

#36 - 11/08/2013 05:27 PM - Vadim Gindin

Behavior of other tests is identical.

About points 1 and 2 I just don't understand this logic.. May be we stay current implementation as incomplete and set TODO there?

#37 - 11/11/2013 11:18 AM - Greg Shah

Please check your tests into the testcases/uast/ project. Make a testcases/uast/last_event/ subdirectory. I will need to review your code for the editing phase case, in order to give you an answer to issue 1. Please let me know what the recreate sequence is for issue 1.

Issue 2 is not a problem. We have known deviations in P2J as compared to some 4GL installations, where the 4GL reports PFx and we report Fx instead. This can differ by terminal type, so it is a bit complicated and it is not entirely wrong anyway. Ignore this deviation for now.

Constantin can help you with issue 3.

#38 - 11/11/2013 12:55 PM - Vadim Gindin

I've committed test cases. Rev #1065. See editing.p for issue 1. Recreation sequence is simple - run editing.p - and see message area right after start. You will see ? symbol. And run converted class Editing.java - you will see PF1 there right after start (no actions needed).

#39 - 11/11/2013 01:08 PM - Greg Shah

When you run code like message my-var + " some text"., the result will be a ? when my-var is the unknown value. In this case, I expect that last-event:label and/or last-event:event-type start off initialized to the unknown value in the 4GL. This makes some sense, since there has not been any events processed at that time. You'll have to change the code to test/output for unknown value.

Generally, any use of unknown value in an expression with operators will cause the expression to evaluate to unknown value. It is important not to get fooled by this behavior.

#40 - 11/12/2013 03:10 AM - Constantin Asofiei

Vadim Gindin wrote:

3) How to test converted procedures that requires two parts: client and server, sessProc and test? I couldn't test such cases.

I guess you are referring to the sockets and async call cases. For sockets, create a starter program which would look like this:

```
message "enter 1 for socket-server and 2 for socket-client:" update i as int.
if i = 1 then run socket-server.p.
if i = 2 then run socket-client.p.
```

You will have to convert this program and the socket client/server programs all at once. After this, you will first start the socket server in a P2J client and secondly the socket client in another P2J client.

I'll get back to you about appservers.

#41 - 11/12/2013 12:06 PM - Vadim Gindin

Good idea.

I tried to run such procedure (converted) and faced with problems:

1) It don't run external procedure. In the Progress I should specify the directory with external procedures to PROPATH variable. How will it work in Java environment?
Behavior of this procedure is simple if I input any number - it reloads UI to the same start page (input of i).

2) I tried to debug (from Eclipse) this cases and also got a problem: clients don't start and outputs the error to log (see client_master_2813.log). I met this error earlier from time to time - not constantly. Could you advice me how to get rid of it?

#42 - 11/12/2013 12:07 PM - Vadim Gindin

- File client_master_2813.log added

Here the file with error log

#43 - 11/12/2013 12:24 PM - Constantin Asofiei

> 1) It don't run external procedure. In the Progress I should specify the directory with external procedures to PROPATH variable. How will it work in Java environment?
> Behavior of this procedure is simple if I input any number - it reloads UI to the same start page (input of i).

This is because RUN statement can't find the target; have you converted all the programs (runner, server and client) in the same conversion run? Remember to clean the testcases/uast before doing this. Also, check the generated name_map.xml if all programs are mapped.

> 2) I tried to debug (from Eclipse) this cases and also got a problem: clients don't start and outputs the error to log (see client_master_2813.log). I met this error earlier from time to time - not constantly. Could you advice me how to get rid of it?

To debug the client in Eclipse, you need to start it in Swing mode; add this client node (or replace it if it exists) to your client.xml:

```
<node type="client">
   <!-- other nodes... ->
   ...

   <client>
      <driver type="swing_chui_frame"/>
      <chui rows="24"/>
      <chui columns="80"/>
      <chui background="0x000000"/>
      <chui foreground="0xFFA500"/>
      <chui selection="0x0000FF"/>
      <chui fontname="monospaced"/>
      <chui fontsize="12"/>
   </client>
</node>
```

After this, the client UI will be pure-java, and you will be able to debug it.
Also, you need to make sure that both the P2J client and the P2J server are running on the same bytecode. The InvalidClassCastException problem is probably because you've started the server and client from different locations. For me, the setup is this way:

1. the P2J project in Eclipse has the src folder added as a source folder
2. if I need native support, I make sure the .so libraries are in the build/lib/ folder and set the library path accordingly (in the debug configuration)
3. no P2J jars will ever be in build/lib
4. both the server and the client are started from Eclipse.

This way, both the client and the server work on the same bytecode and Eclipse is pretty smart to do hot-code replacement pretty well.

#44 - 11/12/2013 05:26 PM - Vadim Gindin

- File server-socket-error.log added

Thanks. Good advices. I set all correctly. The problem was in paths. Mappings contains path dependent of "testcases/uast", but my procedures are in subfolder. So I had to write "/lastevent/server-socket.p" in source procedure.

After that there is an error in server-socket.p and at this moment I didn't find out reason. See server-socket-err.log in attachments.

#45 - 11/13/2013 01:34 AM - Constantin Asofiei

Vadim Gindin wrote:

> After that there is an error in server-socket.p and at this moment I didn't find out reason. See server-socket-err.log in attachments.

memptr implementation in P2J uses native code. Is the libp2o.so created and in the library path?

#46 - 11/13/2013 05:00 PM - Vadim Gindin

No, It wasn't in the java.library.path and you are right - that was the reason of error with memptr. Thank you.

Going further I ran converted client-server scripts and got following results:

client-socket-out:

```
We are in socket client
Connected OK
Start event handling
                                                                        We are in READ-RESPONSE event
procedure, readProc
14 bytes read

cString
----------------------------------------------------------------
SERVER - hello
                                                          send back data to the server
Freeing up resources
Finished
```

and server-socket-out:

```
We are in the socket server
Start event handling
Enabled connections: yes
                                                                        We are in CONNECT event proced
ure, connProc
server before:  RETURN
                                                                        server after:  RETURN

clientMessage
----------------------------------------------------------------
Client - hi
FIRST CONNECTION RECEIVED
** Server-Socket is still connected. Cannot DELETE. (10096)
```

As you can see LAST-EVENT:LABEL before and after is "RETURN". It don't conforms to Progress results described earlier (before: "CONNECT" and after - "READ-RESPONSE"). It is expectable because of proposed implementation (Keyboard.keyLabel()). "CONNECT" and "READ-RESPONSE" aren't names of key events. That are names of other events. I should implement this case somehow dependent of EVENT-TYPE.

I tried to check EVENT-TYPE in windev01, but when I tried to run server-socket.p - Progress became idle. Could you check it? By the way I successfully ran client-socket.p that connected to server-socket and finished. It means that the problem in my script, but it works earlier..

Another one problem is the error "** Server-Socket is still connected. Cannot DELETE. (10096)" but obviously this problem do not refer to LAST-EVENT:LABEL.

#47 - 11/14/2013 05:21 AM - Constantin Asofiei

Vadim Gindin wrote:

> As you can see LAST-EVENT:LABEL before and after is "RETURN". It don't conforms to Progress results described earlier (before: "CONNECT" and after - "READ-RESPONSE"). It is expectable because of proposed implementation (Keyboard.keyLabel()). "CONNECT" and "READ-RESPONSE" aren't names of key events. That are names of other events. I should implement this case somehow dependent of EVENT-TYPE.

Please let me know where you have the two tests on windev01, so I can check (or commit them to the testcases project). BTW, KeyReader.getLabel will be used for this case too, you just need to call a KeyReader.setLabelWorker somewhere in the SocketImpl.readResponseEvent and SocketListenerImpl.connectEvent. Don't know how you do it know, but I think you need to push the label set by a key press in the P2J client to the server (and save it in the KeyReader class), instead of KeyReader pulling the LABEL from the P2J Client.

More, you can't rely on KeyReader.setLabel because the LABEL attribute is read-only for LAST-EVENT. You need to annotate these two setters in KeyReader this way:

```
   @LegacyAttribute(name = "LABEL", setter = true, ignored = true)
```

to make it read-only and also make their implementation like this:

```
   public static void setLabel(character label)
   {
      handle.readOnlyError(asHandle(), "LABEL");
   }
```

#48 - 11/14/2013 11:33 AM - Vadim Gindin

Thank you Constantin! Test cases are already committed to bazaar (testcases/uast/lastevent/server-socket.p and testcases/uast/lastevent/client-socket.p). I tried to run them form my home folder on windev01 (C:/Users/vig/work/). I have write-permission only in this folder as I know.

#49 - 11/14/2013 12:23 PM - Constantin Asofiei

The tests work for me on windev01 (maybe was just a glitch). But, what I would like you to do is to try to simplify the socket-related tests, and also check the LAST-EVENT:LABEL attribute in more than one location (beginning, middle, end of procedure).

#50 - 11/15/2013 08:04 AM - Vadim Gindin

As you advised I ran scripts from the command line on windev01. It lets my edit procedures corresponding to you last comments. I tried to make them simpler. I removed unnecessary code from the READ-RESPONSE and CONNECT procedures and add EVENT-TYPE attribute. Here are changed procedures and results:

server-socket.p:

```
OUTPUT TO server-socket-out.

DEFINE VARIABLE serverSocket AS HANDLE.
DEFINE VARIABLE aOk AS LOGICAL.
DEFINE VARIABLE serverWriteBuffer AS MEMPTR.
DEFINE VARIABLE serverReadBuffer AS MEMPTR.
DEFINE VARIABLE serverMessage AS CHAR.

/* Marshal data to write */
SET-SIZE(serverWriteBuffer) = 64.
SET-SIZE(serverReadBuffer) = 64.
serverMessage = "SERVER - hello".
PUT-STRING(serverWriteBuffer,1) = serverMessage.

/* Create socket */
MESSAGE "Socket server started - " LAST-EVENT:LABEL "[" LAST-EVENT:EVENT-TYPE "]".
CREATE SERVER-SOCKET serverSocket.
MESSAGE "Start event handling - " LAST-EVENT:LABEL "[" LAST-EVENT:EVENT-TYPE "]".

/* Enable connections */
serverSocket:SET-CONNECT-PROCEDURE( "connProc").
aOk = serverSocket:ENABLE-CONNECTIONS( "-S 3363").
MESSAGE "Enabled connections:" aOk " - " LAST-EVENT:LABEL "[" LAST-EVENT:EVENT-TYPE "]".
IF NOT aOk THEN
RETURN.

/* This WAIT-FOR completes after the first connection */
WAIT-FOR CONNECT OF serverSocket.
MESSAGE "FIRST CONNECTION RECEIVED - " LAST-EVENT:LABEL "[" LAST-EVENT:EVENT-TYPE "]".

serverSocket:DISABLE-CONNECTIONS().
DELETE OBJECT serverSocket.
MESSAGE "Finished - " LAST-EVENT:LABEL "[" LAST-EVENT:EVENT-TYPE "]".

/* Connection procedure for server socket */
PROCEDURE connProc.
   /*Socket recieved as parameter*/
   DEFINE INPUT PARAMETER hSocket AS HANDLE.

   MESSAGE "connProc started - " LAST-EVENT:LABEL "[" LAST-EVENT:EVENT-TYPE "]".

   hSocket:WRITE (serverWriteBuffer, 1, LENGTH(serverMessage)).
   MESSAGE hSocket:BYTES-WRITTEN "bytes written - " LAST-EVENT:LABEL "[" LAST-EVENT:EVENT-TYPE "]".

   /* Wait for response from client */
   WAIT-FOR READ-RESPONSE OF hSocket.
   MESSAGE "read-response event became - " LAST-EVENT:LABEL  "[" LAST-EVENT:EVENT-TYPE "]".

   hsocket:READ(serverReadBuffer,1,hsocket:GET-BYTES-AVAILABLE()).
   MESSAGE "clientMessage: " GET-STRING(serverReadBuffer,1) ". connProc ended - " LAST-EVENT:LABEL "[" LAST-EV
ENT:EVENT-TYPE "]".
```

```
END.
```

and client-socket.p:

```
OUTPUT TO client-socket-out.

DEFINE VARIABLE clientSocket AS HANDLE.
DEFINE VARIABLE acknowledge AS LOGICAL.
DEFINE VARIABLE clientReadBuffer AS MEMPTR.
DEFINE VARIABLE clientWriteBuffer AS MEMPTR.
DEFINE VARIABLE cString AS CHARACTER.
DEFINE VARIABLE clientMessage AS CHARACTER.

SET-SIZE(clientReadBuffer) = 64.
SET-SIZE(clientWriteBuffer) = 64.
clientMessage = "Client - hi".

MESSAGE "Client started - " LAST-EVENT:LABEL "[" LAST-EVENT:EVENT-TYPE "]".
CREATE SOCKET clientSocket.
clientSocket:CONNECT ("-H localhost -S 3363").
IF clientSocket:CONNECTED() THEN
   MESSAGE "Connected OK" LAST-EVENT:LABEL "[" LAST-EVENT:EVENT-TYPE "]".
ELSE DO:
   MESSAGE "Could not connect" LAST-EVENT:LABEL "[" LAST-EVENT:EVENT-TYPE "]".
   RETURN.
END.

/* Do READ-RESPONSE event handling */
MESSAGE "Start event handling" LAST-EVENT:LABEL "[" LAST-EVENT:EVENT-TYPE "]".
clientSocket:SET-READ-RESPONSE-PROCEDURE( "readProc").
/* This WAIT-FOR completes after the first data reception */
WAIT-FOR READ-RESPONSE OF clientSocket.

MESSAGE "Freeing up resources" LAST-EVENT:LABEL "[" LAST-EVENT:EVENT-TYPE "]".
clientSocket:DISCONNECT().
DELETE OBJECT clientSocket.
MESSAGE "Client finished" LAST-EVENT:LABEL "[" LAST-EVENT:EVENT-TYPE "]".

/* Read procedure for socket */
PROCEDURE readProc.
   DEFINE VARIABLE clientReadBuffer AS MEMPTR.

   MESSAGE "readResponseProc started - " LAST-EVENT:LABEL "[" LAST-EVENT:EVENT-TYPE "]" .
   SET-SIZE(clientReadBuffer) = 64.
   clientSocket:READ(clientReadBuffer,1,clientSocket:GET-BYTES-AVAILABLE()).
   MESSAGE clientSocket:BYTES-READ "bytes read" LAST-EVENT:LABEL "[" LAST-EVENT:EVENT-TYPE "]".

   /* send data to server*/
   MESSAGE "send back " GET-STRING(clientReadBuffer,1) "to server - " LAST-EVENT:LABEL "[" LAST-EVENT:EVENT-TY
PE "]".
   PUT-STRING(clientWriteBuffer,1) = clientMessage.
   clientSocket:WRITE (clientWriteBuffer,1,LENGTH(clientMessage)).

   MESSAGE "readResponseProc ended - " LAST-EVENT:LABEL "[" LAST-EVENT:EVENT-TYPE "]".

END PROCEDURE.
```

The results.

server-socket-out:

```
Socket server started -   [ ? ]
Start event handling -   [ ? ]
Enabled connections: yes  -   [ ? ]
connProc started -  ENTER [ PROGRESS ]
14 bytes written -  ENTER [ PROGRESS ]
read-response event became -  ENTER [ PROGRESS ]
```

```
clientMessage:  Client - hi . connProc ended -  ENTER [ PROGRESS ]
FIRST CONNECTION RECEIVED -  ENTER [ KEYPRESS ]
Finished -  ENTER [ KEYPRESS ]
```

and client-socket-out:

```
Client started -   [ ? ]
Connected OK  [ ? ]
Start event handling  [ ? ]
readResponseProc started -  ENTER [ PROGRESS ]
14 bytes read ENTER [ PROGRESS ]
send back  SERVER - hello to server -  ENTER [ KEYPRESS ]
readResponseProc ended -  ENTER [ KEYPRESS ]
Freeing up resources ENTER [ KEYPRESS ]
Client finished ENTER [ KEYPRESS ]
```

As you can see the LABEL always is "ENTER" (btw not RETURN). These results also differs from result of run from Client Procedure Editor posted earlier. There was "CONNECT" and "READ-RESPONSE" correspondingly. What is the right way. You wrote at the start of this task that the right way is to run procedures only from the command line. Is it true for Windows too?

#51 - 11/15/2013 09:36 AM - Constantin Asofiei

I think your approach of using MESSAGE to debug LAST-EVENT:LABEL is unfortunate in this case. This will lead the program to wait for key-presses from you; your inconsistent findings are related to the fact that at some point your tests need your input and the server-side events are pushed to the global event queue in an undetermined order (you can say they are pushed async). After changing the tests like this (no more message statements):

- client-socket.p

```
DEFINE VARIABLE clientSocket AS HANDLE.
DEFINE VARIABLE acknowledge AS LOGICAL.
DEFINE VARIABLE clientReadBuffer AS MEMPTR.
DEFINE VARIABLE clientWriteBuffer AS MEMPTR.
DEFINE VARIABLE cString AS CHARACTER.
DEFINE VARIABLE clientMessage AS CHARACTER.

def var ch as char.
form ch format "x(50)"  with down frame f1.
function show returns int (input i  as int).
   ch = string(i) + ": " + last-event:event-type + " " + last-event:label.
   display ch with frame f1.
   down with frame f1.
end.
```

```
SET-SIZE(clientReadBuffer) = 64.
SET-SIZE(clientWriteBuffer) = 64.
clientMessage = "Client - hi".

CREATE SOCKET clientSocket.
clientSocket:CONNECT ("-H localhost -S 3363").

/* Do READ-RESPONSE event handling */
clientSocket:SET-READ-RESPONSE-PROCEDURE( "readProc").
/* This WAIT-FOR completes after the first data reception */
WAIT-FOR READ-RESPONSE OF clientSocket.

clientSocket:DISCONNECT().
DELETE OBJECT clientSocket.

/* Read procedure for socket */
PROCEDURE readProc.
   DEFINE VARIABLE clientReadBuffer AS MEMPTR.

   show(1).
   SET-SIZE(clientReadBuffer) = 64.
   clientSocket:READ(clientReadBuffer,1,clientSocket:GET-BYTES-AVAILABLE()).
   show(2).
   /* send data to server*/
   PUT-STRING(clientWriteBuffer,1) = clientMessage.
   clientSocket:WRITE (clientWriteBuffer,1,LENGTH(clientMessage)).
   show(3).
END PROCEDURE.
```

- server-socket.p

```
DEFINE VARIABLE serverSocket AS HANDLE.
DEFINE VARIABLE aOk AS LOGICAL.
DEFINE VARIABLE serverWriteBuffer AS MEMPTR.
DEFINE VARIABLE serverReadBuffer AS MEMPTR.
DEFINE VARIABLE serverMessage AS CHAR.

def var ch as char.
form ch format "x(50)" with down frame f1.
function show returns int(input i as int).
   ch = string(i) + ": " + last-event:event-type + " " + last-event:label.
   display ch with frame f1.
   down with frame f1.
end.

/* Marshal data to write */
SET-SIZE(serverWriteBuffer) = 64.
SET-SIZE(serverReadBuffer) = 64.
serverMessage = "SERVER - hello".
PUT-STRING(serverWriteBuffer,1) = serverMessage.

/* Create socket */
CREATE SERVER-SOCKET serverSocket.

/* Enable connections */
serverSocket:SET-CONNECT-PROCEDURE( "connProc").
aOk = serverSocket:ENABLE-CONNECTIONS( "-S 3363").
IF NOT aOk THEN
RETURN.

/* This WAIT-FOR completes after the first connection */
WAIT-FOR CONNECT OF serverSocket.

serverSocket:DISABLE-CONNECTIONS().
DELETE OBJECT serverSocket.

/* Connection procedure for server socket */
PROCEDURE connProc.
   /*Socket recieved as parameter*/
      DEFINE INPUT PARAMETER hSocket AS HANDLE.
   show(1).
```

```
   hSocket:WRITE (serverWriteBuffer, 1, LENGTH(serverMessage)).
   show(2).
   /* Wait for response from client */
   WAIT-FOR READ-RESPONSE OF hSocket.
   show(3).
   hsocket:READ(serverReadBuffer,1,hsocket:GET-BYTES-AVAILABLE()).
   show(4).
end.
```

produces this output:

- server-socket.p

```
┌──────────────────────────────────────────────────┐
│ch                                                │
├──────────────────────────────────────────────────┤
│1: PROGRESS CONNECT                               │
│2: PROGRESS CONNECT                               │
│3: PROGRESS READ-RESPONSE                         │
│4: PROGRESS READ-RESPONSE                         │
```

- client-socket.p

```
┌──────────────────────────────────────────────────┐
│ch                                                │
├──────────────────────────────────────────────────┤
│1: PROGRESS READ-RESPONSE                         │
│2: PROGRESS READ-RESPONSE                         │
│3: PROGRESS READ-RESPONSE                         │
```

#52 - 11/15/2013 05:16 PM - Vadim Gindin

- File vig_upd20131115a.zip added

I'm posting update but I have open questions.

I've added label field to WorkArea to hold the label. So I need the logic of KeyReader.getLabel() method to decide when to return Keyboard.getLabel() or when to return label field value? I doubt that posted implementation of Keyboard.getLabel() is correct because of the thought, that the next key event (right after connect event) will not be shown, because of label field already contains the value.

And again the Progress tells that I should do it depending on EVENT-TYPE:

```
For EVENT-TYPE="PROGRESS", this attribute returns the same high-level event name returned by the FUNCTION attr
ibute unless the Progress event is triggered by a key press. In this case it returns the key label of the key
that triggered the event.
```

The difficulty is that EVENT-TYPE is not implemented. How to do it?

Another difficulty is how to determine when the Progress event is triggered by a key press?

As I understand your previous comments we manually set label for special events (like socket events) manually instead of determining the EVENT-TYPE. Is it right?

So please review the update.

#53 - 11/18/2013 02:10 AM - Constantin Asofiei

About implementing the EVENT-TYPE attribute:
1. in methods_attributes.rules, you need to treat the kw_evt_type in two places: once in the static resource reference section (how you already did) and once in rule starting at line 1019 (where the non-static references are treated). Also, you need to set the hwrap variable to LastEvent (in both places).
2. you need to add an API definition to CommonLastEvent interface. Don't forget to add the LegacyAttribute annotation.
3. you don't need to add a setter in KeyReader class, as the attribute is already read-only. More, for internal P2J usage, add a setter with no LegacyAttribute annotation and which sets this attribute.

About the LABEL attribute implementation; as I was saying, I think it's cleaner for the P2J client to push to the server the LABEL attribute, as needed. Looks like this can be done by using the LAST-KEY implementation, which is managed by the KeyReader class on server-side. We can't just interrogate the workArea.obtain().lastkey in the KeyReader.getLabel implementation, as we will not know how to distinguish between i.e. a READ-RESPONSE and a real key. Instead, add a WorkArea.label and a WorkArea.eventType fields, which will be set either explicitly (by some KeyReader.setLabel and KeyReader.setEventType APIs) or implicitly, when the KeyReader.setLastKey is called.

More, I think there is something special when the LAST-KEY is set to something other than -1 and i.e. a CONNECT event is processed. If the server-socket.p program at note 51 is modified so that an UPDATE is executed before the CREATE SERVER-SOCKET statement:

```
...
function show returns int(input i as int).
```

```
   ch = string(i) + ": " + last-event:event-type + " " + last-event:label + " " + string(last-key).
   display ch with frame f1.
   down with frame f1.
end.
...
def var i as int.
update i.
message string(last-key) last-event:label.
/* Create socket */
CREATE SERVER-SOCKET serverSocket.
...
```

the output will be like this:

```
┌──────────────────────────────────────────────────┐
│ch                                                │
│──────────────────────────────────────────────────│
│1: PROGRESS RETURN -1                             │
│2: PROGRESS RETURN -1                             │
│3: PROGRESS READ-RESPONSE -1                      │
│4: PROGRESS READ-RESPONSE -1                      │
```

If the client-socket.p is modified in a similar way, it will output:

```
┌──────────────────────────────────────────────────┐
│ch                                                │
│──────────────────────────────────────────────────│
│1: PROGRESS RETURN -1                             │
│2: PROGRESS RETURN -1                             │
│3: PROGRESS RETURN -1                             │
```

The trailing -1 is the LAST-KEY value. This I think suggests that, when server-side events are encountered, 4GL works like this:

1. if the LAST-KEY is set to something other than -1, it will set it to -1 but will not touch the LABEL
2. if the LAST-KEY is set to -1, it will set the LABEL attribute.

This I think should explain the inconsistent behaviour for the two programs, when there were lots of MESSAGE statements.
Let me know if you have any questions.

#54 - 11/18/2013 05:39 PM - Vadim Gindin

- File vig_upd20131118a.zip added

Constantin Asofiei wrote:

> About implementing the EVENT-TYPE attribute:
> 1. in methods_attributes.rules, you need to treat the kw_evt_type in two places: once in the static resource reference section (how you already did) and once in rule starting at line 1019 (where the non-static references are treated). Also, you need to set the hwrap variable to LastEvent (in both places).

Could you describe me how the rule works? I added but don't understand why there. By the way I'm only know about LAST-EVENT: usage not by some dynamic possible usage of EVENT-TYPE.. Or there is possible to assign LAST-EVENT handle to some var?

> 2. you need to add an API definition to CommonLastEvent interface. Don't forget to add the LegacyAttribute annotation.

What is necessity of it? This interface is not explicitly mentioned in method_attributes.rules.

> 3. you don't need to add a setter in KeyReader class, as the attribute is already read-only. More, for internal P2J usage, add a setter with no LegacyAttribute annotation and which sets this attribute.
>
> About the LABEL attribute implementation; as I was saying, I think it's cleaner for the P2J client to push to the server the LABEL attribute, as needed. Looks like this can be done by using the LAST-KEY implementation, which is managed by the KeyReader class on server-side. We can't just interrogate the workArea.obtain().lastkey in the KeyReader.getLabel implementation, as we will not know how to distinguish between i.e. a READ-RESPONSE and a real key. Instead, add a WorkArea.label and a WorkArea.eventType fields, which will be set either explicitly (by some KeyReader.setLabel and KeyReader.setEventType APIs) or implicitly, when the KeyReader.setLastKey is called.

Please review update. Do I correctly changed the code?

> More, I think there is something special when the LAST-KEY is set to something other than -1 and i.e. a CONNECT event is processed. If the server-socket.p program at note 51 is modified so that an UPDATE is executed before the CREATE SERVER-SOCKET statement:
> [...]
> the output will be like this:
> [...]
> If the client-socket.p is modified in a similar way, it will output:
> [...]

I found some description in the javadoc to ThinClient.readKey() about the value -1

```
-1 if timed out, otherwise the event code
```

Interesting how it works in case of server events.

> The trailing -1 is the LAST-KEY value. This I think suggests that, when server-side events are encountered, 4GL works like this:
>
> 1. if the LAST-KEY is set to something other than -1, it will set it to -1 but will not touch the LABEL
> 2. if the LAST-KEY is set to -1, it will set the LABEL attribute.
>
> This I think should explain the inconsistent behaviour for the two programs, when there were lots of MESSAGE statements.
> Let me know if you have any questions.

Where should I place this logic? To the setLastKey and readKey methods (as methods that set WorkArea.lastkey field)?

More, the ThinClient.readKey contains two last lines:

```
    // this will maintain LASTKEY too, which is required!
     // target widget is not important
     KeyInput key = (KeyInput) waitForEvent(seconds, false, false);

     return checkForChoose(seconds, key);
```

waitForEvent returns ServerEvent in case of server events. But the checkForChoose method extracts only keyCode from event. So at this place an information about class of event (is it server or client) is loosed.
If we'll save this information here We will be able to set LAST-EVENT:LABEL dynamically and independent of concrete server event. For example:

```
ServerEvent event = ...;
KeyReader.setLabelWorker(event.getName());
```

instead of manual setting in each server event case (connect, read-response and possible others).
Is it make sense?

#55 - 11/18/2013 07:10 PM - Greg Shah

> Or there is possible to assign LAST-EVENT handle to some var?

Exactly: my-handle-var = LAST-EVENT. message my-handle-var:LABEL.

> What is necessity of it? This interface is not explicitly mentioned in method_attributes.rules.

We use dynamic proxies to properly implement the Progress error handling. That means that all Java methods that replace 4GL attributes or 4GL methods, must be defined in an interface.

In addition, the LegacyAttribute annotation allows our runtime code to dynamically "find" the Java methods that replace 4GL attributes.

In regard to your other questions, I think answers will have to wait a couple of days until Constantin is available.

#56 - 11/20/2013 10:53 AM - Constantin Asofiei

Sorry, I was mistaken in saying you need to treat EVENT-TYPE in the "global" attribute/method handling too... is enough to treat it in the methods_attributes.rules's LAST-EVENT section (which you've already did), as this covers the variable case too. So, please remove the methods_attributes.rules:1024 line change. Also, you should add some tests to check if h:LABEL and h:EVENT-TYPE work OK at runtime and conversion.

> 2. you need to add an API definition to CommonLastEvent interface. Don't forget to add the LegacyAttribute annotation.
>
> What is necessity of it? This interface is not explicitly mentioned in method_attributes.rules.

To find out who defines the resource, you need to look at the return type of the handle.unwrapLastEvent API (for this case). Thus, you need to add an API to the CommonLastEvent interface; else, an h:event-type call (when h references the LAST-EVENT resource) will not work.

> Please review update. Do I correctly changed the code?

You are on the correct path:
KeyReader

1. in KeyReader.setLastKey, the EVENT-TYPE I think it should be set to KEYPRESS (but see bellow about CHOOSE).
2. initially, the WorkArea.eventType should be null; make sure to test what are the default values for these attributes, in a test like this:

```
if last-event:event-type <> ? then message "last-event:event-type initially must be unknown".
if last-event:label <> "" then message "last-event:label initially must be empty string".
```

   Remember to run it from the command line.
3. you don't need a legacy-style setter for EVENT-TYPE (I refer to KeyReader.setEventType), as this attribute is already read-only and the CommonLastEvent interface does not (and should not) define the setter. Explicit legacy-style setters (with the ignore = true annotation in place) are added only when the attribute is used by more than one resource, and for certain resources is read-only while for others is writeable. If we are forced to add a setter for a read-only attribute, this is why we need to mark it with ignore = true and explicitly call handle.readOnlyError.

SocketImpl and SocketListenerImpl

1. after you change the LABEL, you need to set the event-type too.

   > The trailing -1 is the LAST-KEY value. This I think suggests that, when server-side events are encountered, 4GL works like this:
   > ...
   >
   > Where should I place this logic? To the setLastKey and readKey methods (as methods that set WorkArea.lastkey field)?

KeyReader.setLabelWorker is the place, as we will need this only for server-side events (at least this is how it looks now); eventually, add a boolean parameter to know what to do: force the WorkArea.label change or just set WorkArea.lastKey to -1.
BTW, you need to change the javadoc for KeyReader.setLabelWorker to something like this (you currently refer to Not intended for using in conversion process. and Intended for internal usage only for example in socket events. which may lead the reader to incorrect path):

```
* Used internally by P2J to set the @WorkArea#label@ in server-side event cases.
```

> More, the ThinClient.readKey contains two last lines:
> [...]
>
> waitForEvent returns ServerEvent in case of server events.

If you look carefully, you will see that the ThinClient.readKey will not listen for server-side events (this waitForEvent call will never return a server-side

event); this is because the waitForEvent's honorServerEvent parameter is set to false.

> But the checkForChoose

If you mention it, the documentation states that for the CHOOSE event (and others), the EVENT-TYPE is set to PROGRESS and not KEYPRESS; you will need to check if the last-key code is synthetic or not, before deciding to set EVENT-TYPE to PROGRESS or KEYPRESS, in KeyReader.readkey and setLastKey.

> ... method extracts only keyCode from event. So at this place an information about class of event (is it server or client) is loosed.

See above, this assumption is incorrect.

#57 - 11/21/2013 01:17 PM - Vadim Gindin

Constantin Asofiei wrote:

> ..
> you will need to check if the last-key code is synthetic or not ..

How to check it?

#58 - 11/21/2013 01:20 PM - Constantin Asofiei

Vadim Gindin wrote:

> Constantin Asofiei wrote:
>
> > ..
> > you will need to check if the last-key code is synthetic or not ..
>
> How to check it?

Keyboard.java has only a few synthetic codes (prefixed with SE_); this includes server-side events. Also, please check how EVENT-TYPE and LABEL are reported for these events (via APPLY or UI interaction).

#59 - 11/21/2013 02:07 PM - Vadim Gindin

It works different than we expected:
ENTRY, LEAVE - KEYPRESS inspite of they are synthetic in Keyboard.java. CHOOSE - "PROGRESS". So we need more complicated criteria probably.

#60 - 11/21/2013 04:43 PM - Vadim Gindin

- File vig_upd20131122a.zip added

Here is next update. If I correctly understand we have the only one open question - about criteria of EVENT-TYPE setting.

#61 - 11/22/2013 07:05 AM - Constantin Asofiei

Two notes about the changes:

- Keyboard.java - unneeded import
- Keyboard.isSynth - make sure the javadoc can be seen nicely in html (i.e. add a <p> if you need to split rows).

Otherwise, the changes look good. I guess the following need to be finished?

- the server-side quirk I mentioned in note 53
- ENTRY/LEAVE I see they are still marked as synthetic (so they will have the EVENT-TYPE set to PROGRESS); do you have tests for all synthetic events?
- changes for the PROCEDURE-COMPLETE event need to be added to the AsyncRequestImpl.execute method, before the event is posted (similar to how you've done it for the socket events).

#62 - 11/22/2013 04:17 PM - Vadim Gindin

Constantin Asofiei wrote:

> ..
>
> - Keyboard.isSynth - make sure the javadoc can be seen nicely in html (i.e. add a <p> if you need to split rows).

Thank I will know. But there is just one string..)

> ..
>
> - ENTRY/LEAVE I see they are still marked as synthetic (so they will have the EVENT-TYPE set to PROGRESS); do you have tests for all synthetic events?

I'v tested them.
The main thing is that "NEXT-FRAME", "PREV-FRAME", "ENTRY", "LEAVE", "CHOOSE" - are KEY FUNCTION, but we implement only KEY LABEL and EVENT-TYPE. Here are the tests.

ENTRY,LEAVE,CHOOSE - one of previous tests

NEXT_FRAME (ESC-TAB key)

```
def button b1 label "b1".
def button b2 label "b2".
```

```
display b1 with frame f1 side-labels.
display b2 with frame f2 side-labels.

message "Press "  KBLABEL("NEXT-FRAME").

on next-frame of b1
  message "next " LAST-EVEN:LABEL " - " last-event:function " - [" last-event:event-type "]".
on prev-frame of b2
  message "prev " LAST-EVENT:LABEL " - " last-event:function " - [" last-event:event-type "]".

on choose of b1
  apply "NEXT-FRAME" to b1 in frame f1.

on choose of b2
  apply "PREV-FRAME" to b2 in frame f2.

enable all with frame f1.
enable all with frame f2.

wait-for go of frame f1 focus b1.
```

and DEFAULT_ACTION (called by DELETE button) and VALUE_CHANGED

```
def var hist-date as date format "99/99/9999"
  view-as combo-box
  list-items 07/04/1776, 07/11/1969, 09/10/1993.

def frame main-frame
   hist-date
   with no-labels title "Historic Events".

on default-action /* value-changed*/of hist-date
  message "last-event. " last-event:label " - " last-event:function " [ " last-event:event-type "]".

enable hist-date with frame main-frame.

apply "value-changed" to hist-date in frame main-frame.

wait-for window-close of current-window.
```

I found that EVENT-TYPE for Key Functions: CHOOSE, VALUE_CHANGED, DEFAULT_ACTION is PROGRESS. In other cases EVENT-TYPE is KEYPRESS (ENTRY, LEAVE, NEXT-FRAME, PREV-FRAME). So synthetic is a bad criteria to set EVENT_TYPE. What do you think?

#63 - 11/23/2013 03:29 AM - Constantin Asofiei

Vadim Gindin wrote:

> I found that EVENT-TYPE for Key Functions: CHOOSE, VALUE_CHANGED, DEFAULT_ACTION is PROGRESS. In other cases EVENT-TYPE is KEYPRESS (ENTRY, LEAVE, NEXT-FRAME, PREV-FRAME). So synthetic is a bad criteria to set EVENT_TYPE. What do you think?

Then, make it more granular: rename the new Keyboard API to something like "isProgressEventType" which will return true only if the EVENT-TYPE should be set to PROGRESS.

#64 - 11/25/2013 12:09 PM - Vadim Gindin

- File vig_upd20131125a.zip added

Here is next update. LABEL is implemented correctly but EVENT-TYPE is not implemented correctly.
When user presses the button (RETURN key code) EVENT-TYPE="KEYPRESS" but should be "PROGRESS". Current implementation operates only lastkey - key code. But pressing RETURN (one key code) can have different EVENT-TYPE. It depends on circumstances: pressing a button - EVENT-TYPE=PROGRESS, in other cases - it should be "KEYPRESS" (for example editing text in editor field).

#65 - 11/25/2013 12:23 PM - Vadim Gindin

By the way. When user presses the button (RETURN key code), key function is CHOOSE (EVENT-TYPE:PROGRESS). In other cases key function is RETURN - same as key code. lastkey that we use in Keyboard.isPGEvent() is key code not a key function. Do I wrong?

If I'm right. May be we should implement key function first, and then make an implementation of EVENT-TYPE depending on an implementation of key function? Alternatively we can stay this implementation at this moment and set aside key function implementation to a future.

#66 - 11/26/2013 04:33 AM - Constantin Asofiei

About the update:

- AsyncRequestImpl is missing history entry
- have you tested the READ-RESPONSE/CONNECT events?

About the CHOOSE event:

- is the key function you are referring something 4GL-specific?
- please post a note with your test for the CHOOSE event.

#67 - 11/26/2013 11:27 AM - Vadim Gindin

Constantin Asofiei wrote:

> About the update:

- AsyncRequestImpl is missing history entry

Thank you, I've added.

- have you tested the READ-RESPONSE/CONNECT events?

Yes I did. Current implementation works correctly in this case.

About the CHOOSE event:

- is the key function you are referring something 4GL-specific?

Key Function is some form of abstraction of term "action". There is a key code, key label and key function (action). Key code is just integer code of key pressed (-2 for ex.) . Key label is just symbolic name of a key pressed (F4 for example) and Key function is the logic action that should be ran for this key (END-ERROR). By the way key function can be changed in the configuration. I.e. we can change the keys that execute certain key function (action). More over, there can be several keys for one key function and these keys can differ for Windows and Unix cases.

- please post a note with your test for the CHOOSE event.

I wrote separate test for this case. Here it is:

```
def var edit as char view-as editor size 10 by 2 no-undo.
def button somebut label "Press me".

display
  edit at 2 skip (0.1)
  somebut at 5
with frame f side-labels.

on choose of somebut
  run msgwatch.

on return, any-printable, entry, leave of edit
  run msgwatch.

enable all with frame f.
wait-for go of frame f focus edit.

procedure msgwatch.
  message last-event:label + " " + last-event:function + "[" + last-event:event-type + "]".
```

This procedure shows a frame with editor field and button. We have listeners for events of editor and button, that ran common procedure showing the last-event attributes: label, function and event-type (in specified order).

Here are results of tests:
1) The moment right after UI appeared on the screen. Focus in editor field.
ENTRY ENTRY [PROGRESS]
2) We pressed Enter. Cursor is moved from the first line of editor to the second line.
RETURN RETURN [KEYPRESS]
3) We pressed Tab. Focus moved from the editor to the button.
TAB LEAVE [KEYPRESS]
4) We pressed Enter (on the button)
RETURN CHOOSE [PROGRESS].

What we have? We pressed Enter 2 times: on the editor field and on the button and got different results:

```
LABEL=RETURN,
FUNCTION=RETURN (editor field)
```

```
         CHOOSE (button)
EVENT-TYPE=KEYPRESS (editor field - we changed the value)
          =PROGRESS (button)
```

More over we have interesting case - the moment right after ran.
1) The LABEL had an ENTRY value (unexpectedly). No keys was pressed at this moment..
2) Key function was ENTRY but the EVENT-TYPE=PROGRESS (!). It was also unexpected. It means that EVENT-TYPE also can be different for ENTRY key function. In this case it is PROGRESS probably because of function ENTRY was called without pressing any key. But if we add another one field before editor and we'll come to editor field from that field by pressing TAB, than EVENT-TYPE will be KEYPRESS for key function ENTRY. LABEL will be TAB.

#68 - 11/27/2013 07:39 AM - Constantin Asofiei

Vadim Gindin wrote:

- have you tested the READ-RESPONSE/CONNECT events?

Yes I did. Current implementation works correctly in this case.

I'm a little confused; I don't see the logic for the case described at note 53 in your update. Is that no longer needed?

Key Function is some form of abstraction of term "action".

I see now, you are referring to LAST-EVENT:FUNCTION. I think this better translates to: this is the name of the event raised when the user pressed a certain key.

> More over we have interesting case - the moment right after ran.
> 1) The LABEL had an ENTRY value (unexpectedly). No keys was pressed at this moment..

I think this is because by default an ENTRY event is applied to the widget.

> 2) Key function was ENTRY but the EVENT-TYPE=PROGRESS (!). It was also unexpected. It means that EVENT-TYPE also can be different for ENTRY key function. In this case it is PROGRESS probably because of function ENTRY was called without pressing any key. But if we add another one field before editor and we'll come to editor field from that field by pressing TAB, than EVENT-TYPE will be KEYPRESS for key function ENTRY. LABEL will be TAB.

I think the solution will be to push to the server the name of the last event (i.e. the function of the last pressed key); this needs to be done in a couple of places:

1. ThinClient.processProgressEvent - use the action determined at line 12812
2. ThinClient.checkForChoose - if this is a Button, explicitly push Keyboard.SE_CHOOSE

To push it to the server, add a new field in ClientState, which will be initialized in ThinClient.getChanges with the name of the last event (saved by the one of the two methods above). The server will read this in LogicalTerminal.applyChanges.

#69 - 11/27/2013 05:30 PM - Vadim Gindin

> I think the solution will be to push to the server the name of the last event (i.e. the function of the last pressed key); this needs to be done in a couple of places:
>
> 1. ThinClient.processProgressEvent - use the action determined at line 12812
> 2. ThinClient.checkForChoose - if this is a Button, explicitly push Keyboard.SE_CHOOSE
>
> To push it to the server, add a new field in ClientState, which will be initialized in ThinClient.getChanges with the name of the last event (saved by the one of the two methods above). The server will read this in LogicalTerminal.applyChanges.

I tried to implement but at this moment it don't work.

#70 - 11/28/2013 01:35 AM - Constantin Asofiei

Vadim Gindin wrote:

> I tried to implement but at this moment it don't work.

Well, then tell/show us where you get stuck/why it doesn't work. And if you have other ideas to implement this, I'll be glad to hear them.

#71 - 11/28/2013 01:26 PM - Vadim Gindin

- File vig_upd20131129a.zip added

Here is the code. The problem I can see is that a process that got changes from the client and calls LogicalTerminal.applyChanges and writes it to WorkArea is not the same as the process that reads WorkArea label during execution of converted procedure. So the changes are not visible to that process..

#72 - 11/29/2013 04:14 AM - Constantin Asofiei

The state synchronization code (ClientState and ServerState) is currently executed in the Reader and Writer threads; there is no problem here, as on the i.e. server side the Reader and Writer threads are using the same LogicalTerminal instance as the Conversation thread (for a certain context). Same on client side: the Reader and Writer threads are using the correct ThinClient instance.

#73 - 11/29/2013 04:16 AM - Constantin Asofiei

PS: please upgrade your project and merge the update with the latest P2J revision.

#74 - 11/29/2013 03:24 PM - Vadim Gindin

I'm sorry, I need to describe the problem more concretely. It is in a fact that we have 2 different WorkArea instances: in the first we save a label, got from the LogicatTerminal.applyChanges(), and the second instance - the converted class is getting the label from (during execution right after start).

Why are there 2 different instances? When should I save the label to WorkArea?

#75 - 11/30/2013 08:29 AM - Constantin Asofiei

Are you referring to the fact that you want to access the KeyReader$WorkArea class from LogicalTerminal.applyChanges, the state synchronizer code on server side? If so, this is not currently possible; the applyChanges code (both on client and server side) should save the state in the ThinClient and LogicalTerminal instances respectively; the Reader/Writer threads don't have a context, thus is not possible to access context-local data.

#76 - 12/03/2013 02:14 PM - Vadim Gindin

It helped. Current problems.

1) It's hard to summarize behavior of Progress.
1a) We know, that LABEL=ENTRY (high level event name) right after start procedure, LABEL=RETURN in both cases: the user pressed enter on the button and in the editor field.
1b) EVENT-TYPE=PROGRESS in cases: first entry and when the user presses enter on the button.
in other cases EVENT-TYPE=KEYPRESS
1c) We don't implement LAST-EVENT:FUNCTION yet. Isn't it?

2) We saved lastActionName in the LogicalTerminal.applyChanges as you adviced. Where to read this info to WorkArea? I tried 2 variants:
2a) KeyReader.WorkArea.initialValue - the place where WorkArea is initializing right after it created.
For this case 1a works, 1b - don't work, 1c - don't work but don't expected.
2b) KeyReader.WorkArea.getLabel() - the place where LABEL is read.
For this case 1a don't work - the values are similiar to FUNCTION values: (ENTRY, LEAVE, CHOOSE), 1b - don't work: for LEAVE and ENTRY in such case it shows PROGRESS but should show KEYPRESS, 1c - don't work but don't expected.

Should the implementation cover FUNCTION attribute? It may require more testing.
Or it should only cover LABEL and EVENT-TYPE

to implement EVENT-TYPE we need an algorithm that Progress uses to define PROGRESS or KEYPRESS for concrete high-level event (CHOOSE, ENTRY, LEAVE...). It depends on situation.

checkForChoose() is not called

Could you advice a better place where we can load the action from LogicalTerminal to WorkArea?

#77 - 12/04/2013 09:09 AM - Constantin Asofiei

1. in LT.applyChanges, where do you save lastActionName? I expect it be in some LT instance field, which later on is read by KeyReader.
2. about FUNCTION attribute: although we don't have conversion support for it, doesn't the key "action" computed in processProgressEvent act like the FUNCTION attribute? If not, then look into how processProgressEvent can be changed so that the correct key "function" can be computed in processProgressEvent (to be sent back to the server).
3. why are you saying "checkForChoose() is not called"? It was added there for a reason, do you mean you can't find a testcase so that this code is reached?

#78 - 12/04/2013 09:30 AM - Vadim Gindin

Constantin Asofiei wrote:

1. in LT.applyChanges, where do you save lastActionName? I expect it be in some LT instance field, which later on is read by KeyReader.

Yes I did exactly like you're saying. It works. The problems is where is to read it from LT field to KeyReader.WorkArea. That is what I talked about in previous note.

1. about FUNCTION attribute: although we don't have conversion support for it, doesn't the key "action" computed in processProgressEvent act like the FUNCTION attribute? If not, then look into how processProgressEvent can be changed so that the correct key "function" can be computed in processProgressEvent (to be sent back to the server).

It works similar to FUNCTION, but it needs to make additional testing to find it out precisely. Also I was expecting that action attribute will work as LABEL attribute.. Should we make support of FUNCTION attribute in this task? I tested in common LABEL and EVENT-TYPE attributes

1. why are you saying "checkForChoose() is not called"? It was added there for a reason, do you mean you can't find a testcase so that this code is reached?

In my testcase this method is not called (note #67).

#79 - 12/04/2013 09:57 AM - Constantin Asofiei

> Also I was expecting that action attribute will work as LABEL attribute..

No, we need the action (aka function) to determine how to map the event-type for a label... in the end, to fully-support it, looks like we will need all three; depending on the action/label combinations, we should be able to map event-type properly (i.e. if key is RETURN and function is CHOOSE, then event-type is PROGRESS; is possible maybe to check only the function, if the function is for a progress event, then event-type is progress). The conversion changes for FUNCTION should be pretty simple, so go ahead and add them, as we are already on our way to add runtime support for it. Also, rename the "action" to "function", to not make things more ambigous.

1. why are you saying "checkForChoose() is not called"? It was added there for a reason, do you mean you can't find a testcase so that this code is reached?

In my testcase this method is not called (note #67).

Then, look for a testcase which reaches this code.

#80 - 12/11/2013 06:45 PM - Vadim Gindin

- File vig_upd20131211a.zip added

Here is following update and several open questions mostly about the FUNCTION implementation.

1) Proposed update works for most frequent events like CHOOSE, LEAVE, ENTRY and so on.

2) Msgupdate.p

```
def button rd label "red".
def var i as int init 0.
def var h as handle.

display rd with frame f side-labels.
on choose of rd
do:
    h = LAST-EVENT.
    message i " before: "h:LABEL "-" h:function " [" h:event-type "] " update i.
    message i " after: " h:LABEL "-" h:function " [" h:event-type "] " .
end.
enable all with frame f.
wait-for go of frame f.
```

Different results:

```
Progress
   43 after: RETURN-GO[KEYPRESS]
Java
   43 after: RETURN-LEAVE[KEYPRESS]
```

The reason is in following. There events chain appears during key press processing: RETURN->GO->LEAVE
All three events are really appeared: Enter was pressed (RETURN), we are going from the field (LEAVE) and we are processing entered value (GO).
It is really not obvious what event is more important here: LEAVE or GO. Probably GO.

But our algorithm is following. In the ThinClient.processProgressEvent() we are using the last event from this chain and outputting LEAVE. I suspect that the Progress uses some events priorities to understand what event to use for the function. This case is simple. Same problem exists in other cases.

3) Developer events - is a local case. The event chain here could be following: RETURN->LEAVE->U1.
Progress ignores developer events (U1-U10) and I made the custom solution for this case.

4) Editing.p

```
def var i as char.
update i with frame f1 editing:
  message "1)" last-event:label " " last-event:function " [" last-event:event-type "] " .
  readkey.
  message "2)" last-event:label " " last-event:function " [" last-event:event-type "] " .
```

```
  apply lastkey.
  message "3)" last-event:label " " last-event:function " [" last-event:event-type "] " .
end.
```

In this case we are reading input symbols one by one and outputting last-event attributes between the input and applying concrete symbol.

Here are results.

```
Progress
  3  3  [KEYPRESS]
Java
  3 VALUE-CHANGED [KEYPRESS]
```

The reason of this difference is in the problem of caching functionKey in LogicalTerminal. From the one side we are reading symbols one by one in a KeyReader.readKey() and calculating FUNCTION there. From the other side LogicalTerminal has cached value of the FUNCTION in its field. I don't understand when should I clear this field. By the way Java behavior seems more logical than Progress.

5) Choose.p case outputs
ENTRY ENTRY [PROGRESS]
right after start as I wrote earlier in the note #67

first ENTRY is a LAST-EVENT:LABEL. After some time I think that its a Progress bug. ENTRY is not a valid value for the LABEL attribute, it is a high level event.

Current java implementations shows
? ENTRY [PROGRESS]
and it seems more correct that Progress variant.

6) About checkForChoose(). Please look at the place in it where I paste my assignment. I've checked all my tests - this assignment is not executed in any of them.

#81 - 12/12/2013 10:10 AM - Constantin Asofiei

About the update:
methods_attributes.rules
- header is merged wrong

KeyBoard

```
   /* lowest developer event */
   public static final int DE_LOWEST = 551; // eventCode(U1); don't work because of WorkArea is null

   /* highest developer event */
   public static final int DE_HIGHEST = 560; // eventCode(U10); don't work because of WorkArea is null
```

- I think you meant to use keyCode, which requires WorkArea. You should move these two inside WorkArea, and initialize them using WorkArea.findKey.

I'll take a look at the rest of your findings tomorrow morning; in the mean time, continue with #2018.

#82 - 12/13/2013 07:28 AM - Constantin Asofiei

2) Msgupdate.p

I think this test is a little too far ahead of our understanding of the LAST-EVENT behaviour. For MESSAGE ... UPDATE statement, the correct order of the events is RETURN:LEAVE:GO:, as shown by the following test:

```
def var i as int.
def var evts as char init "".

on go anywhere do:
   evts = evts + "GO:".
end.

on leave anywhere do:
   evts = evts + "LEAVE:".
end.

on "return" anywhere do:
   evts = evts + "RETURN:".
end.

message "enter i:" update i.

if evts <> "RETURN:LEAVE:GO:" then message "event order is incorrect" evts.
```

Even in your case, where the MESSAGE ... UPDATE is nested in a trigger, the order of the events fired by this statement is the same.
Now, about why P2J shows LEAVE on after: I think this is because we don't fire a GO in ThinClient.tinyInput; more, P2J doesn't fire any of the

RETURN/LEAVE/GO events during MESSAGE ... UPDATE.

4) Editing.p

The problem is because VALUE-CHANGED should be ignored, when setting functionKey in ThinClient.processProgressEvent(). Doing this only if KeyEvent.isSpecial() is false should work, but I think ThinClient.processProgressEvent() is too late. readkey sets the LAST-EVENT:FUNCTION immediately, not when the event is applied...

5) Choose.p case outputs

Please make sure all tests are in repository (what I posted and yours). Also, change your tests to something like this:

```
def var edit as char view-as editor size 10 by 2 no-undo.
def button somebut label "Press me".
def var j as int.
j = 1.

display edit at 2 skip (0.1) somebut at 5 with frame f side-labels.

on choose of somebut run msgwatch.

on return, any-printable, entry, leave of edit run msgwatch.

enable all with frame f.
wait-for go of frame f focus edit.

procedure msgwatch.
    put screen row j col 20 focus:type + " " +  last-event:label + " " + last-event:function + "[" + last-even
t:event-type + "]".
 j = j + 1.
end.
```

to avoid key presses for debug purposes (as they may interfere).
More, for some reason, if this line:

```
on return, any-printable, entry, leave of edit run msgwatch.
```

is changed to:

```
on return, any-printable, entry, leave anywhere run msgwatch.
```

button doesn't have CHOOSE function and PROGRESS event-type, it has RETURN function and KEYPRESS event type.

first ENTRY is a LAST-EVENT:LABEL. After some time I think that its a Progress bug. ENTRY is not a valid value for the LABEL attribute, it is a high level event.

I think this is how 4GL sends the default ENTRY event to the first focusable and enabled widget... we should implement it this way.

6) About checkForChoose(). Please look at the place in it where I paste my assignment. I've checked all my tests - this assignment is not executed in any of them.

I think the location is OK, but see this testcase:

```
def button btn label "btn".
def var j as int.
j = 1.
```

```
form btn with frame f1.

on "choose" of btn do:
   put screen row j col 20 last-event:label.
   put screen row j col 30 last-event:function.
   put screen row j col 40 last-event:event-type.

   j = j + 1.
end.

update btn with frame f1 editing:
   readkey.

   put screen row j col 20 last-event:label.
   put screen row j col 30 last-event:function.
   put screen row j col 40 last-event:event-type.
   j = j + 1.

   apply lastkey.
end.
```

Conclusions:

1. make sure the initial values for all these attributes are OK. Check the following cases:
   - right after program startup
   - right after wait-for loop starts (at the beginning first caught event)
   - anything else you can think of
2. build a list of the cases you've covered (in tests and code changes) and from those, build a list of which cases are not working properly. Make sure your debug code and test scenario do not interfere. Start the list from simple tests to complex tests.

I want to have a summary of the behaviour we found and the implementation state of these findings, as this task has proven more complex than on initial assessment.

#83 - 12/16/2013 07:24 PM - Vadim Gindin

Here are the test cases. I rewrote it to make it simpler and independent of key presses or other events.

1) Readkey

```
message "before: " LAST-EVENT:LABEL "-" last-event:function " [" last-event:event-type "] ".
readkey pause 5.
message "after: " LAST-EVENT:LABEL "-" last-event:function " [" last-event:event-type "] ".
```

Right after start
PG: - [?]
Java - Ctrl-@ [KEYPRESS]

When user presses the button - test works correctly.

2) Prompt-for

```
def var x as int init 3.
def var b as char.
def var a as char.

b = LAST-EVENT:LABEL + "-" + last-event:function + " [" + last-event:event-type + "] " .
prompt-for x.
a = LAST-EVENT:LABEL + "-" + last-event:function + " [" + last-event:event-type + "] " .

message "x_init=" x "X_specified=" input x.
message "b=" b "a= " a.
```

PG: RETURN-LEAVE [KEYPRESS]
Java: RETURN-GO [KEYPRESS]

3) Anyprintable

```
def var edit as char view-as editor size 10 by 2 no-undo.

def var txt0 as char format "x(75)" no-undo.
display "Feel free to move around..." view-as text
  edit at 2 skip (0.1)
  with frame f side-labels.

display txt0 label "last-event:label function (type)"
with frame report 4 down centered title "Messages".

on any-printable of edit
  run msgwatch. /* This procedure uses last-event */

enable all with frame f.

wait-for go of frame f focus edit.

/* outputs LAST-EVENT info */
procedure msgwatch.
 txt0 = LAST-EVENT:LABEL + " " + LAST-EVENT:FUNCTION + " [" + LAST-EVENT:EVENT-TYPE + "] ".
 display txt0 with frame report.
 down with frame report.
end procedure.
```

PG and Java works identically.

4) checkforchoose

```
def button btn label "btn".
def var j as int.
j = 1.

form btn with frame f1.

on "choose" of btn do:
```

```
   message "choose".

   put screen row j col 20 last-event:label.
   put screen row j col 30 last-event:function.
   put screen row j col 40 last-event:event-type.

   j = j + 1.
end.

update btn with frame f1 editing:

   message "update".

   readkey.

   put screen row j col 20 last-event:label.
   put screen row j col 30 last-event:function.
   put screen row j col 40 last-event:event-type.
   j = j + 1.

   apply lastkey.
end.
```

PG: on backspace there was no messages
TAB TAB KEYPRESS user pressed TAB
TAB CHOOSE PROGRESS user pressed RETURN
CHOOSE CHOOSE PROGRESS user pressed RETURN (!) // label=CHOOSE
JAVA: on backspace - it outputs LABEL=BACKSPACE
TAB ENTRY KEYPRESS user pressed tab
TAB TAB KEYPRESS user pressed tab
RETURN TAB KEYPRESS user pressed RETURN
RETURN CHOOSE PROGRESS user pressed RETURN

"Caches" previous values of attributes.
Fully unimplemented. See ThinClient.checkForChoose method.

5) Editing phrase

```
def var i as char.

update i with frame f1 editing:
  message "1:" last-event:label " " last-event:function " [" last-event:event-type + "] " .
  readkey.
  message "2:" last-event:label " " last-event:function " [" last-event:event-type + "] " .
  apply lastkey.
  message "3:" last-event:label " " last-event:function " [" last-event:event-type + "] " .
end.
```

PG:
- [? (right after start)
4 4 [KEYPRESS]
Java:
? ENTRY [KEYPRESS] (right after start)
4 VALUE-CHANGED [PROGRESS]

6) Before after

```
def var fi as char format "x(50)" label "Fill-in" no-undo.
def button b label "button for focus".

display fi at 2
        b at 10 with frame f side-labels.

on leave of fi
do:
  put screen row 10 col 1 "inside leave".
```

```
  put screen row 11 col 1 LAST-EVENT:LABEL.
  put screen row 12 col 1 LAST-EVENT:FUNCTION .
  put screen row 13 col 1 last-event:event-type.
end.

message 'before leave: ' LAST-EVENT:LABEL " " LAST-EVENT:FUNCTION " [" last-event:event-type "] ".
apply "leave" to fi in frame f.
message 'after leave: ' LAST-EVENT:LABEL " " LAST-EVENT:FUNCTION " [" last-event:event-type "] ".

message 'before U4: ' LAST-EVENT:LABEL " " LAST-EVENT:FUNCTION " [" last-event:event-type "] " .
apply "U4" to fi in frame f.
message 'after U4: ' LAST-EVENT:LABEL " " LAST-EVENT:FUNCTION " [" last-event:event-type "] ".

on U4 of fi
do:
  message 'inside U4: ' LAST-EVENT:LABEL " " LAST-EVENT:FUNCTION " [" last-event:event-type "] ".
end.

enable all with frame f.

wait-for go of frame f focus fi.
```

PG: (right after start)
inside leave

?

before leave: [?]
after leave: [?]

Java: (right after start)
inside leave:
?
LEAVE
KEYPRESS

before leave: ? CTRL-@ [KEYPRESS]
after leave: ? LEAVE [KEYPRESS]

7) Developer events

```
def var fi as char format "x(50)" label "Fill-in" no-undo.

display "Look at messages area" view-as text
  fi skip
  with frame f side-labels.

on U2 of fi
do:
  run msgwatch.
  display "U2 have triggered" with frame report.
end.

apply "U2" to fi in frame f.

enable all with frame f.

wait-for go of frame f focus fi.

/* outputs LAST-EVENT info */
procedure msgwatch.
 message LAST-EVENT:LABEL + " " + LAST-EVENT:FUNCTION + " [" + LAST-EVENT:EVENT-TYPE + "]".
end procedure.
```

PG and Java works identically.

8) Widget key events

```
def var msg_watcher as char format "x(50)" no-undo.
def var fi as char format "x(50)" label "Fill-in" no-undo.
def var edit as char view-as editor size 10 by 2 no-undo.
def button exit label "Go to Exit".
def button test label "Test".

def var txt0 as char format "x(75)" no-undo.
display "Feel free to move around..." view-as text
  skip fi skip (0.1)
  edit at 2 skip (0.1)
  test at 10
  exit at 50
  with frame f side-labels.

on tab, any-printable, go of edit
  run msgwatch. /* This procedure uses last-event */

on entry, leave, go, return, choose of edit
  run msgwatch.

on choose of test
  run msgwatch.

enable all with frame f.

wait-for go of frame f focus fi.

/* outputs LAST-EVENT info */
procedure msgwatch.
 message LAST-EVENT:LABEL + " " + LAST-EVENT:FUNCTION + " [" + LAST-EVENT:EVENT-TYPE + "]".
end procedure.
```

PG and Java works fully identical.

9) Message .. view-as alert-box

```
def button rd label "red".
def var i as int init 67.
def var j as int init 1.
def var txt as char.

display rd with frame f side-labels.

on entry of rd do :
    run msgwatch.
    message " focus " view-as alert-box.
    run msgwatch.
end.

procedure msgwatch.
  txt = last-event:label + "-" + last-event:function + "[" + last-event:event-type + "]".
  put screen row j col 20 txt.
  j = j + 1.
end.

enable all with frame f.
wait-for go of frame f.
```

PG: (right after start)
ENTRY-ENTRY[PROGRESS]
ENTRY-ENTRY[PROGRESS]
Java: infinite loop - couldn't find msgwatch internal procedure...

10) Message .. update

```
def var  i as int.
def var evts as char init "".

on go anywhere do:
  evts = evts + "GO:".
end.

on leave anywhere do:
  evts = evts + "LEAVE:".
end.

on "return" anywhere do:
  evts = evts + "RETURN:".
end.

message "enter i:" update i.

if evts <> "RETURN:LEAVE:GO:" then message "order is incorrect " evts.
else message "ok".
```

PG:
ok
Java:
order is incorrect

In java evts is empty string - no triggers was invoked...

11) Nested statements

```
def var i as int.
def var j as int.

form i with frame f.

on 1 of i
  message "trigger '1': " LAST-EVENT:LABEL "-" last-event:function " [" last-event:event-type "] ".

message "before: " LAST-EVENT:LABEL "-" last-event:function " [" last-event:event-type "] ".
update i with frame f.
message "after: " LAST-EVENT:LABEL "-" last-event:function  " [" last-event:event-type "] ".

enable all with frame f.
wait-for go of frame f.
```

PG:
- [?] (right after start)
RETURN - GO [KEYPRESS] (after user inputs a number and presses enter)
Java:
? - Ctrl-@ [KEYPRESS] (right after start)
RETURN - LEAVE [KEYPRESS] (after user inputs a number and presses enter)

12) VALUE-CHANGED

```
def var hist-date as date format "99/99/9999"
  view-as combo-box
  list-items 07/04/1776, 07/11/1969, 09/10/1993.

def frame main-frame
   hist-date
   with no-labels title "Historic Events".

on value-changed of hist-date
  message "last-event. " last-event:label " - " last-event:function " [ " last-event:event-type "]".

enable hist-date with frame main-frame.
wait-for window-close of current-window.
```

PG:
CURSOR-DOWN - VALUE-CHANGED [PROGRESS]
Java:
RETURN - VALUE-CHANGE [PROGRESS]

--- Server side tests and sessions tests will be posted tomorrow.

Common problems:
1) startup values for all attributes. The problem become more complex than we expected. It seems that initial ENTRY event is only processed. LEAVE for example is not processed by Progress as initial and is not outputted.
2) Ctrl-@ (action=0) - probably some garbage value.
3) GO vs LEAVE functions.
4) CURSOR-DOWN vs RETURN labels.
5) CheckForChoose - completely new testcase
6) VALUE-CHANGED vs 4 (or any alphanum) in editing phrase. Could not be just ignored because of testcase 12

Widget key events and developer events work correctly.

#84 - 12/17/2013 06:15 PM - Vadim Gindin

13) Asynchronous calls

```
def var ha as handle.

procedure proc1.
  message "bla" last-event:label "-" last-event:function " [" last-event:event-type "] ".
end.

pause 3.

run test.p on server session asynchronous set ha event-procedure "proc1".

pause 5.

wait-for procedure-complete of ha.
```

PG: CTRL-SHIFT-ESC-HELP-KEY - ? [KEYPRESS]
Java: ? - Ctrl-@ [KEYPRESS]

If we will delete pause 3 then results will be:
PG: - [?]
Java: ? - Ctrl-@ [KEYPRESS]

To make this procedure runnable I had to correct two NPE in com.goldencode.p2j.util.AsyncRequestImpl.run():
Throwable cause = result null ? null : result.getError(); and
modes = result null ? null : result.getModes();

14) sockets. After some moment procedures started to ran with error:

server-socket.p

```
OUTPUT TO server-socket-out.

DEFINE VARIABLE serverSocket AS HANDLE.
```

```
DEFINE VARIABLE aOk AS LOGICAL.
DEFINE VARIABLE serverWriteBuffer AS MEMPTR.
DEFINE VARIABLE serverReadBuffer AS MEMPTR.
DEFINE VARIABLE serverMessage AS CHAR.
MESSAGE "We are in the socket server".

CREATE SERVER-SOCKET serverSocket.
/* Marshal data to write */
SET-SIZE(serverWriteBuffer) = 64.
SET-SIZE(serverReadBuffer) = 64.
serverMessage = "SERVER - hello".
PUT-STRING(serverWriteBuffer,1) = serverMessage.
MESSAGE "Start event handling".

serverSocket:SET-CONNECT-PROCEDURE( "connProc").
aOk = serverSocket:ENABLE-CONNECTIONS( "-S 3363").
MESSAGE "Enabled connections:" aOk.
IF NOT aOk THEN
RETURN.

/* This WAIT-FOR completes after the first connection */
WAIT-FOR CONNECT OF serverSocket.
MESSAGE "FIRST CONNECTION RECEIVED".

serverSocket:DISABLE-CONNECTIONS().
DELETE OBJECT serverSocket.
MESSAGE "Finished".

/* Connection procedure for server socket */
PROCEDURE connProc.
   /*Socket recieved as parameter*/
   DEFINE INPUT PARAMETER hSocket AS HANDLE.
   MESSAGE "We are in CONNECT event procedure, connProc".
   hSocket:WRITE (serverWriteBuffer, 1, LENGTH(serverMessage)).
   MESSAGE hSocket:BYTES-WRITTEN "bytes written".
   /* Wait for response from client */
   MESSAGE "server before: " LAST-EVENT:LABEL "-" last-event:function "[" last-event:event-type "]".
   WAIT-FOR READ-RESPONSE OF hSocket.
   MESSAGE "server after: " LAST-EVENT:LABEL "-" last-event:function "[" last-event:event-type "]".
   hsocket:READ(serverReadBuffer,1,hsocket:GET-BYTES-AVAILABLE()).
   DEFINE VARIABLE clientMessage AS CHAR.
   clientMessage = GET-STRING(serverReadBuffer,1).
   DISPLAY clientMessage FORMAT "x(64)".
END.
```

and client-socket.p

```
OUTPUT TO client-socket-out.

DEFINE VARIABLE clientSocket AS HANDLE.
DEFINE VARIABLE acknowledge AS LOGICAL.
DEFINE VARIABLE clientReadBuffer AS MEMPTR.
DEFINE VARIABLE clientWriteBuffer AS MEMPTR.
DEFINE VARIABLE cString AS CHARACTER.
DEFINE VARIABLE clientMessage AS CHARACTER.
MESSAGE "We are in socket client".

CREATE SOCKET clientSocket.
clientSocket:CONNECT ("-H localhost -S 3363").
IF clientSocket:CONNECTED() THEN
   MESSAGE "Connected OK".
ELSE DO:
   MESSAGE "Could not connect".
   RETURN.
END.

/* Do READ-RESPONSE event handling */
MESSAGE "Start event handling".
clientSocket:SET-READ-RESPONSE-PROCEDURE( "readProc").
/* This WAIT-FOR completes after the first data reception */
WAIT-FOR READ-RESPONSE OF clientSocket.
```

```
MESSAGE "Freeing up resources".
clientSocket:DISCONNECT().
DELETE OBJECT clientSocket.
MESSAGE "Finished".

/* Read procedure for socket */
PROCEDURE readProc.
   DEFINE VARIABLE clientReadBuffer AS MEMPTR.
   MESSAGE "We are in READ-RESPONSE event procedure, readProc".
   SET-SIZE(clientReadBuffer) = 64.
   SET-SIZE(clientWriteBuffer) = 64.
   clientSocket:READ(clientReadBuffer,1,clientSocket:GET-BYTES-AVAILABLE()).

   PAUSE 5.
   MESSAGE "client. after read: " LAST-EVENT:LABEL "-" last-event:function "[" last-event:event-type "]".

   /*In the socket procedure SELF will point to the handle of the socket for
   which this procedure is set */
   MESSAGE clientSocket:BYTES-READ "bytes read".
   /*Unmarshal data*/
   cString = GET-STRING(clientReadBuffer,1).
   DISPLAY cString FORMAT "x(64)".
   /* send data to server*/
   MESSAGE "send back data to the server".
   clientMessage = "Client - hi".
   PUT-STRING(clientWriteBuffer,1) = clientMessage.

   PAUSE 5.
   MESSAGE "client. before write: " LAST-EVENT:LABEL "-" last-event:function "[" last-event:event-type "]".

   clientSocket:WRITE (clientWriteBuffer,1,LENGTH(clientMessage)).
END PROCEDURE.
```

```
дек 18, 2013 4:56:48 AM com.goldencode.p2j.security.AssociatedThread run
SEVERE: null
java.lang.NullPointerException
    at com.goldencode.p2j.util.MethodInvoker.invoke(MethodInvoker.java:76)
    at com.goldencode.p2j.net.Dispatcher.processInbound(Dispatcher.java:694)
    at com.goldencode.p2j.net.Dispatcher.dispatch(Dispatcher.java:809)
    at com.goldencode.p2j.net.Dispatcher$DispatcherStub.run(Dispatcher.java:1621)
    at java.lang.Thread.run(Thread.java:744)
```

I didn't find out why. MethodInvoker contains method=null and tries to invoke it.

--
What to do with it..?

#85 - 12/17/2013 06:59 PM - Vadim Gindin

I've committed all tests for LAST-EVENT I have. rev #1075.

#86 - 12/18/2013 04:32 AM - Constantin Asofiei

A quick note about the MethodInvoker problems: some static proxies got bad after the handle resource ID changes, I'm working on fixing them.

#87 - 12/18/2013 08:20 AM - Constantin Asofiei

Notes about your findings:

1. please merge your update and post it here.
2. you need to hard code the expected values in the tests, something like (for your first scenario):

```
if (last-event:label <> "")
   then message "initial last-event:label should be empty but is" last-event:label.
if (last-event:function <> "")
   then message "initial last-event:function should be empty but is" last-event:function.
... and so on
```

3. Although posting the tests makes it easier to see what you are testing, when I mentioned a set of rules, I meant something like:
   - case 1: external procedure is started, attributes are checked at the begining of the procedure: LABEL and FUNCTION are empty string, EVENT-TYPE is unknown
   - case 2: check the attributes after the i.e. PROMPT-FOR statement: LABEL is RETURN, FUNCTION is GO, EVENT-TYPE is KEYPRESS
   - and so on. I expect this to expand a lot, as new cases are added.
   I think a table will be easier to read, with a column describing the scenario and additional columns for each attribute, with the P2J and 4GL behaviour. You can add a separate comment to maintain this table and, as we discuss it, you can edit it with the new findings (and you can use ~~strikethrough~~ for maintaining a history).
4. for scenario 6) (and others), you are still using MESSAGE statement. At some point, this will ask you for a key. I suspect this will interfere with your scenario. That's why is best to hard-code the expected results in the test...
5. the NPE fixes for the problems in AsyncRequestImpl.run are OK
6. about the infinite loop caused by "can't find msgwatch internal procedure" - make sure you cleanup the testcases/ project (by running testcases/uast/cleanup.sh) before running conversion. Else, the name_map.cache is reused and some times you get unexpected behaviour, after a previously-converted test program is added new internal procedures.
7. see #2183 for a fix about the MethodInvoker problems. You can integrate it with your update, as I expect for it to pass runtime testing and I will release it once it does.

#88 - 12/23/2013 05:59 PM - Vadim Gindin

Table contains testcases and results for both Progress and Java.

"Progress" column contains expected output. Testcases are modified to output only in case of unexpected values. So Progress procedures do not output anything really (mostly), but column will contain expected attributes values for better understanding. Expected values are also contained in text

of procedures except of couple of it.

"Java" column will contain real output. It will mean errors - values different from expected (in Progress). The column will also contain descriptions of errors in *Italic font*.

Unitialized values of our attrs are: LABEL="", FUNCTION="", EVENT-TYPE=?

#	Testcase	Progress	Java
1.	**CheckForChoose - CHOOSE event**		
	/* Checking this event during the first press at start of procedure		
	(reading and applying key) and during the button pressing */		
	def button btn label "btn".	LABEL=CHOOSE	LABEL=RETURN
	def var j as int.	FUNCTION=CHOOSE	FUNCTION=ENTRY (first press of btn)
	j = 1.		FUNCTION=CHOOSE (other times)
	form btn with frame f1.	EVENT-TYPE=PROGRESS	EVENT-TYPE=KEYPRESS (first press of btn)
	on "choose" of btn do:		EVENT-TYPE=PROGRESS (other times)
			--
	if last-event:label <> "CHOOSE" then		*ERROR: LABEL=RETURN*
	put screen row j col 20 last-event:label.		*ERROR: FUNCTION=ENTRY (first press of btn)*
	if last-event:function <> "CHOOSE" then		*ERROR: EVENT-TYPE=KEYPRESS (first press of btn)*
	put screen row j col 30 last-event:function.		
	if last-event:event-type <> "PROGRESS" then		
	put screen row j col 40 last-event:event-type.		
	j = j + 1.		
	end.		
	/* don't actually understand what this block does */		
	update btn with frame f1 editing:		
	readkey.		
	if last-event:label <> "CHOOSE" then		
	put screen row j col 20 last-event:label.		
	if last-event:function <> "CHOOSE" then		
	put screen row j col 30 last-event:function.		
	if last-event:event-type <> "PROGRESS" then		
	put screen row j col 40 last-event:event-type.		
	j = j + 1.		

	apply lastkey.		
	end.		
2.	**Choose2 - CHOOSE: "anywhere" and "of but", ENTRY, LEAVE**		
	/* Testing difference of triggers subject for CHOOSE event:		
	on choose anywhere and on choose of someButton.		
	Also testing RETURN, ENTRY, LEAVE events */		
	def button but label "Press me".	See triggers	Works identical to Progress
	def var edit as char view-as editor size 10 by 2 no-undo.		
	def var j as int.		
	j = 1.		
	display but at 2 skip (0.1) edit at 5 with frame f side-labels.		
	function check returns int (input type as char, input lab as char,		
	input func as char, input evt as char).		
	if last-event:label <> lab then		
	message type ": wrong label: " last-event:label.		
	if last-event:function <> func then		
	message type ": wrong function: " last-event:function.		
	if last-event:event-type <> evt then		
	message type ": wrong event-type" last-event:event-type.		
	end.		
	/* different results for anywhere and for concrete button */		
	on choose of but check("choose of button", "RETURN", "CHOOSE", "PROGRESS").		
	on choose anywhere check("choose anywhere", "RETURN", "RETURN", "KEYPRESS").		
	on return of edit check("return in edit", "RETURN", "RETURN", "KEYPRESS").		
	on entry of edit check("entry in edit", "TAB", "ENTRY", "KEYPRESS").		
	on leave of edit check("leave in edit", "TAB", "LEAVE", "KEYPRESS").		
	enable all with frame f.		
	wait-for go of frame f focus		

	somebut.		
3.	**Editing phrase**		
	/* Testing attributes between reading and applying		
	stages of editing phrase trigger */		
	def var i as char.	Start values	Outputs (right after start)
	def var evt as char init ?.	LABEL=""	1wrong label: ?
		FUNCTION=""	1wrong function: ENTRY
	update i with frame f1 editing:	EVENT-TYPE=?	1wrong event-type: KEYPRESS
			Press digit 4.
	if last-event:label <> keylabel(lastkey) then	LABEL=4	2wrong function: ENTRY
	message "1wrong label: " last-event:label.	FUNCTION=4	3wrong function: VALUE-CHANGED
	if last-event:function <> keyfunction(lastkey) then	EVENT-TYPE=KEYPRESS	--
	message "1wrong function: " last-event:function.		*ERROR: FUNCTION=VALUE-CHANGED*
	if last-event:event-type <> evt then		*ERROR: LABEL=? (right after start)*
	message "1wrong event-type: " last-event:event-type.		Press enter
			3wrong event-type: PROGRESS
	readkey.		2wrong function: VALUE-CHANGED
			--
	evt = "KEYPRESS".		*ERROR: EVENT-TYPE=PROGRESS*
			ERROR: FUNCTION=VALUE-CHANGED
	if last-event:label <> keylabel(lastkey) then		
	message "2wrong label: " last-event:label.		
	if last-event:function <> keyfunction(lastkey) then		
	message "2wrong function: " last-event:function.		
	if last-event:event-type <> evt then		
	message "2wrong event-type: " last-event:event-type.		
	apply lastkey.		
	if last-event:label <> keylabel(lastkey) then		
	message "3wrong label: " last-event:label.		
	if last-event:function <> keyfunction(lastkey) then		
	message "3wrong function: " last-event:function.		
	if last-event:event-type <> evt then		
	message "3wrong event-type: " last-event:event-type.		

	end.		
4.	**ENTRY event - right after start**		
	/* Testing ENTRY event at the start of procedure.		
	The feature is that LABEL will have ENTRY value		
	that is not in available values for LABEL. It valid for FUNCTION */		
	def var ch as char.	LABEL=ENTRY	Outputs (right after start)
		FUNCTION=ENTRY	right after start : wrong label: ?
	display ch with frame f side-labels.	EVENT-TYPE=PROGRESS	right after start : wrong event-type: KEYPRESS
			--
	function check returns int (input type as char, input lab as char,		*ERROR: LABEL=?*
	input func as char, input evt as char).		*ERROR: EVENT-TYPE=KEYPRESS*
	if last-event:label <> lab then		
	message type ": wrong label: " last-event:label.		
	if last-event:function <> func then		
	message type ": wrong function: " last-event:function.		
	if last-event:event-type <> evt then		
	message type ": wrong event-type" last-event:event-type.		
	end.		
	on entry of ch check("right after start", "ENTRY", "ENTRY", "PROGRESS").		
	enable all with frame f.		
	wait-for go of frame f focus ch.		
5.	**Frames (NEXT-FRAME) - events of moving between frames**		
	/* Procedure displays two buttons each in its own frame.		
	Switching between the frames occuring when user presses ESC-TAB.		
	We are testing attrs during this switching */		
	def button b1 label "b1".	LABEL=ESC-TAB	Works identical to Progress
	def button b2 label "b2".	FUNCTION=NEXT-FRAME	
		EVENT-TYPE=KEYPRESS	
	display b1 with frame f1 side-labels.		
	display b2 with frame f2 side-labels.		
	message "Press " KBLABEL.		

	on next-frame of b1		
	do:		
	if LAST-EVENT:LABEL <> "ESC-TAB" then		
	message "wrong label: " LAST-EVEN:LABEL.		
	if LAST-EVENT:function <> "NEXT-FRAME" then		
	message "wrong function " last-event:function.		
	if last-event:event-type <> "KEYPRESS" then		
	message "wrong event-type " last-event:event-type.		
	end.		
	enable all with frame f1.		
	enable all with frame f2.		
	wait-for go of frame f1 focus b1.		
6.	Check before, after and inside		
	def var fi as char format "x(50)" label "Fill-in" no-undo.		
	def button b label "button for focus".		
	display "Press TAB to enter to the text field" view-as text		
	b skip fi with frame f side-labels.		
	function check returns int (input nm as char, input l as char,		
	input f as char, input t as char).		
	if last-event:label <> l then		
	message nm " - wrong label: " last-event:label.		
	if last-event:function <> f then		
	message nm " - wrong function: " last-event:function.		
	if last-event:event-type <> t then		
	message t " - wrong event-type: " last-event:event-type.		
	end.		
	on choose of b		
	do:		
	check("inside choose", "TAB", "ENTRY", "KEYPRESS").		
	end.		
	/**		
	* we can see that apply does not affect last-event attrs.		
	* they are the same: before,inside, after applying		
	*/		

	on entry of fi		
	do:		
	check("before choose", "TAB", "ENTRY", "KEYPRESS").		
	apply "choose" to b in frame f.		
	check("after choose", "TAB", "ENTRY", "KEYPRESS").		
	end.		
	enable all with frame f.		
	wait-for go of frame f focus b.		
7.	**Developer Events - U1-U10, and apply statement**		
	/* Testing how developer events are using attrs		
	We can only trigger developer event using apply statement.		
	We also test this statement. It do not affecting our attrs */		
	def var fi as char format "x(50)" label "Fill-in" no-undo.	LABEL=""	Outputs right after start:
		FUNCTION=""	unexpected label: ?
	display "Look at messages area" view-as text	EVENT-TYPE=?	unexpected function: CTRL-@
	fi skip with frame f side-labels.		--
			ERROR: LABEL=?
	on U2 of fi		*ERROR: FUNCTION=CTRL-@*
	do:		
	if last-event:label <> "" then		
	message "unexpected label: " last-event:label.		
	if last-event:function <> "" then		
	message "unexpected function: " last-event:function.		
	if last-event:event-type <> ? then		
	message "unexpected event-type: " last-event:event-type.		
	display "U2 have triggered" with frame report.		
	end.		
	apply "U2" to fi in frame f.		
	enable all with frame f.		
	wait-for go of frame f focus fi.		
8.	**Widget and key events - any-printable, tab, entry, leave, return, choose**		
	/* Testing widget events (tab, entry, leave, return, choose)		
	and key events for example when user is typing some text in editor field.		
	Procedure displays 2 edit fields		

	and 2 buttons. */		
	def var msg_watcher as char format "x(50)" no-undo.	see testcase	Works identical to Progress
	def var fi as char format "x(50)" label "Fill-in" no-undo.	For key events attrs	
	def var edit as char view-as editor size 10 by 2 no-undo.	LABEL and FUNCTION always contain	
	def button exit label "Go to Exit".	the typed symbol	
	def button test label "Test".	and EVENT-TYPE=KEYPRESS	
	def var txt0 as char format "x(75)" no-undo.		
	display "Feel free to move around..." view-as text		
	skip fi skip (0.1) edit at 2 skip (0.1)		
	test at 10 exit at 50 with frame f side-labels.		
	/* compares params with last-event:attributes */		
	function trylast returns int (input nm as char, input l as char,		
	input f as char, input e as char).		
	if last-event:label <> l then		
	message nm " wrong label: " last-event:label.		
	if last-event:function <> f then		
	message nm " wrong function: " last-event:function.		
	if last-event:event-type <> e then		
	message nm " wrong event-type: " last-event:event-type.		
	end.		
	on tab /*, any-printable */ of edit		
	trylast("tab from edit", "TAB", "TAB", "KEYPRESS").		
	on any-printable of edit		
	trylast("any-printable of edit", keylabel(lastkey), keyfunction(lastkey), "KEYPRESS").		
	on entry of edit		
	trylast("entry of edit", keylabel(lastkey), "ENTRY", "KEYPRESS").		
	on leave of edit		
	trylast("leave of edit", keylabel(lastkey), "LEAVE", "KEYPRESS").		
	on return of edit		

	trylast("return in edit", "RETURN", "RETURN", "KEYPRESS").		
	on choose of test		
	trylast("choose of button", "RETURN", "CHOOSE", "PROGRESS").		
	enable all with frame f.		
	wait-for go of frame f focus fi.		
9.	**Message .. view-as alert-box**		
	/* Testing how this statement affects the attrs. Just look at attrs		
	before and after call */		
	def button rd label "red".		Still fall into infinite
	def var i as int init 67.		loop during execution when
	def var j as int init 1.		tries to find msgwatch procedure
	def var txt as char.		
	display rd with frame f side-labels.		
	/** message .. view-as alert-box does not affect last-event attrs */		
	run msgwatch.		
	message " focus " view-as alert-box.		
	run msgwatch.		
	procedure msgwatch.		
	if last-event:label <> "" then		
	do:		
	put screen row j col 20 "wrong label:" + last-event:label.		
	j = j + 1.		
	end.		
	if last-event:function <> "" then		
	do:		
	put screen row j col 20 "wrong label:" + last-event:function.		
	j = j + 1.		
	end.		
	if last-event:event-type <> ? then		
	do:		
	put screen row j col 20 "wrong label:" + last-event:event-type.		
	j = j + 1.		
	end.		
	end.		
	enable all with frame f.		

	wait-for go of frame f.			
10.	**Message .. update - checking of events sequence(RETURN, LEAVE, GO)**			
	/* This test only checks the sequence of generated events			
	during "message .. update" statement */			
	def var i as int.	RETURN:LEAVE:GO:	Java sequence is empty	
	def var evts as char init "".			
	/* message .. update statement - events sequence */			
	on go anywhere do:			
	evts = evts + "GO:".			
	end.			
	on leave anywhere do:			
	evts = evts + "LEAVE:".			
	end.			
	on "return" anywhere do:			
	evts = evts + "RETURN:".			
	end.			
	message "enter i:" update i.			
	if evts <> "RETURN:LEAVE:GO:" then message "order is incorrect " evts.			
11.	**Nested events**			
	/* As I remember this test checks how the attrs affected			
	during the nested triggered events */			
	def var i as int.	LABEL=""	LABEL=?	
	def var j as int.	FUNCTION=""	FUNCTION=CTRL-@	
		EVENT-TYPE=?		
	form i with frame f.			
	/* update statement does not behave itself unexpectedly */			
	function check returns int (input nm as char, input l as char,			
	input f as char, input e as char).			
	if last-event:label <> l then			
	message nm "-wrong label: " last-event:label.			
	if last-event:function <> f then			s
	if last-event:event-type <> e then			

	message nm "-wrong label: " last-event:event-type.		
	end.		
	on 1 of i check("trigger 1", "1", "1", "KEYPRESS").		
	check("before", "", "", ?).		
	update i with frame f.		
	/* to finish data entry we have to press enter */		
	check("after", "RETURN", "GO", "KEYPRESS").		
	enable all with frame f.		
	wait-for go of frame f.		
12.	**Readkey**		
	/* Tests attrs before and after READKEY statement */		
	function check returns in (input nm as char, input l as char,	LABEL=""	LABEL=?
	input f as char, input e as char).	FUNCTION=""	FUNCTION=CTRL-@
	if last-event:label <> l then	EVENT-TYPE=?	
	message nm "-wrong label:" last-event:label.		
	if last-event:function <> f then		
	message nm "-wrong function:" last-event:function.		
	if last-event:event-type <> e then		
	message nm "-wrong event-type:" last-event:event-type.		
	end.		
	check("before", "", "", ?).		
	readkey pause 5.		
	check("after", keylabel(lastkey), keyfunction(lastkey), "KEYPRESS").		
13.	**Prompt-for**		
	/* Tests attrs before and after PROMPT-FOR statement */		
	def var x as int init 3.	RETURN-GO[KEYPRESS]	RETURN-LEAVE[KEYPRESS]
	def var b as char.		
	def var a as char.		
	b = LAST-EVENT:LABEL + "-" + last-event:function + "[" + last-event:event-type + "]" .		
	prompt-for x.		
	a = LAST-EVENT:LABEL + "-" + last-event:function + "[" + last-event:event-type + "]" .		
	message "x_init=" x "X_specified="		

	input x.		
	if a <> "RETURN-GO[KEYPRESS]" then message "wrong a: " a.		
	if b <> ? then message "wrong b: " b.		
14.	**VALUE-CHANGED**		
	/* This test displays combo-box field and checks the attrs during		
	the user changes selected value of it */		
	def var msg as char.	CURSOR-DOWN - VALUE-CHANGED[PROGRESS]	RETURN - VALUE-CHANGED[PROGRESS]
	def var hist-date as date format "99/99/9999"		
	view-as combo-box		
	list-items 07/04/1776, 07/11/1969, 09/10/1993.		
	def frame main-frame		
	hist-date with no-labels title "Historic Events".		
	on value-changed of hist-date		
	do:		
	msg = last-event:label + " - " + last-event:function + " [" + last-event:event-type + "]".		
	if msg <> "CURSOR-DOWN - VALUE-CHANGED [PROGRESS]" then message "wrong last-event:" msg.		
	end.		
	enable hist-date with frame main-frame.		
	wait-for window-close of current-window.		
15.	**EVENT-PROCEDURE of ASYNCHRONOUS CALL**		
	/* This test makes asynchronous call of external procedure		
	and checks the attrs during event-procedure execution */		
	def var ha as handle.	LABEL=""	LABEL=?
		FUNCTION=""	FUNCTION=CTRL-@
	/* event-procedure of asynchronous call. Output nothing */	EVENT-TYPE=?	
	procedure proc1.		
	if last-event:label <> "" then message "wrong label: " last-event:label.		

	if last-event:function <> "" then message "wrong function: " last-event:function.		
	if last-event:event-type <> ? then message "wrong event-type: " last-event:event-type.		
	end.		
	run test.p on server session asynchronous set ha event-procedure "proc1".		
	wait-for procedure-complete of ha.		
16.	**ENDKEY processing**		
	/* Tests the attrs in the trigger of ENDKEY */		
	def button but label "quit".	LABEL=PF4	Trigger wasn't called at all
		EVENT-TYPE=KEYPRESS	
	display but with frame f.	FUNCTION=END-ERROR	
	enable all with frame f.		
	on "F4", "PF4" anywhere		
	do:		
	if last-event:label <> "PF4" then		
	message "wrong label:" last-event:label.		
	if last-event:function <> "END-ERROR" then		
	message "wrong function:" last-event:function.		
	if last-event:event-type <> "KEYPRESS" then		
	message "wrong event-type:" last-event:event-type.		
	end.		
17.	**CONNECT, READ-RESPONSE events (client-server interactions)**		
	/* **Client procedure.** Creates client socket and connects read-response	LABEL=READ-RESPONSE	LABEL=READ-RESPONSE
	procedure to it. We are looking at different stages of this procedure */	FUNCTION=READ-RESPONSE	FUNCTION=LEAVE (!)
	DEFINE VARIABLE clientSocket AS HANDLE.	EVENT-TYPE=PROGRESS	EVENT-TYPE=KEYPRESS (!)
	DEFINE VARIABLE acknowledge AS LOGICAL.	LABEL=READ-RESPONSE	LABEL=READ-RESPONSE
	DEFINE VARIABLE clientReadBuffer AS MEMPTR.	FUNCTION=READ-RESPONSE	FUNCTION=LEAVE (!)
	DEFINE VARIABLE clientWriteBuffer AS MEMPTR.	EVENT-TYPE=PROGRESS	EVENT-TYPE=KEYPRESS (!)
	DEFINE VARIABLE cString AS CHARACTER.	LABEL=READ-RESPONSE	LABEL="" (!)
	DEFINE VARIABLE clientMessage AS CHARACTER.	FUNCTION=READ-RESPONSE	FUNCTION=LEAVE (!)
		EVENT-TYPE=PROGRESS	EVENT-TYPE=KEYPRESS (!)
	function CHECKLAST returns int (input type as char, input lab as char,		

	input func as char, input evt as char).		
	if last-event:label <> lab then		
	message type ": wrong label: " last-event:label.		
	if last-event:function <> func then		
	message type ": wrong function: " last-event:function.		
	if last-event:event-type <> evt then		
	message type ": wrong event-type" last-event:event-type.		
	end.		
	SET-SIZE = 64.		
	SET-SIZE = 64.		
	clientMessage = "Client - hi".		
	CREATE SOCKET clientSocket.		
	clientSocket:CONNECT ("-H localhost -S 3363").		
	/* Do READ-RESPONSE event handling */		
	clientSocket:SET-READ-RESPONSE-PROCEDURE.		
	/* This WAIT-FOR completes after the first data reception */		
	WAIT-FOR READ-RESPONSE OF clientSocket.		
	clientSocket:DISCONNECT.		
	DELETE OBJECT clientSocket.		
	/* Read procedure for socket */		
	PROCEDURE readProc.		
	DEFINE VARIABLE clientReadBuffer AS MEMPTR.		
	CHECKLAST.		
	SET-SIZE = 64.		
	clientSocket:READ).		
	CHECKLAST.		
	/* send data to server*/		
	PUT-STRING = clientMessage.		
	clientSocket:WRITE (clientWriteBuffer,1,LENGTH).		
	CHECKLAST.		
	END PROCEDURE.		

	/* **Server procedure**. Defines server socket and connect procedure for it	LABEL=READ-RESPONSE	LABEL=READ-RESPONSE
	We are looking at start, end, after	FUNCTION=READ-RESPONSE	FUNCTION=LEAVE (!)

	waiting and reading stages of it */		
	DEFINE VARIABLE serverSocket AS HANDLE.	EVENT-TYPE=PROGRESS	EVENT-TYPE=KEYPRESS (!)
	DEFINE VARIABLE aOk AS LOGICAL.	LABEL=READ-RESPONSE	LABEL=READ-RESPONSE
	DEFINE VARIABLE serverWriteBuffer AS MEMPTR.	FUNCTION=READ-RESPONSE	FUNCTION=LEAVE (!)
	DEFINE VARIABLE serverReadBuffer AS MEMPTR.	EVENT-TYPE=PROGRESS	EVENT-TYPE=KEYPRESS (!)
	DEFINE VARIABLE serverMessage AS CHAR.	LABEL=READ-RESPONSE	LABEL=READ-RESPONSE
		FUNCTION=READ-RESPONSE	FUNCTION=LEAVE (!)
	function CHECKLAST returns int (input type as char, input lab as char, input func as char, input evt as char).	EVENT-TYPE=PROGRESS	EVENT-TYPE=KEYPRESS (!)
	if last-event:label <> lab then	LABEL=READ-RESPONSE	LABEL=READ-RESPONSE
	message type ": wrong label: " last-event:label.	FUNCTION=READ-RESPONSE	FUNCTION=LEAVE (!)
	if last-event:function <> func then	EVENT-TYPE=PROGRESS	EVENT-TYPE=KEYPRESS (!)
	message type ": wrong function: " last-event:function.		
	if last-event:event-type <> evt then		
	message type ": wrong event-type" last-event:event-type.		
	end.		
	/* Marshal data to write */		
	SET-SIZE = 64.		
	SET-SIZE = 64.		
	serverMessage = "SERVER - hello".		
	PUT-STRING = serverMessage.		
	/* Create socket */		
	CREATE SERVER-SOCKET serverSocket.		
	/* Enable connections */		
	serverSocket:SET-CONNECT-PROCEDURE.		
	aOk = serverSocket:ENABLE-CONNECTIONS.		
	IF NOT aOk THEN		
	RETURN.		
	/* This WAIT-FOR completes after the first connection */		
	WAIT-FOR CONNECT OF serverSocket.		
	serverSocket:DISABLE-CONNECTIONS.		
	DELETE OBJECT serverSocket.		

	/* Connection procedure for server socket */		
	PROCEDURE connProc.		
	/*Socket recieved as parameter*/		
	DEFINE INPUT PARAMETER hSocket AS HANDLE.		
	CHECKLAST.		
	hSocket:WRITE (serverWriteBuffer, 1, LENGTH).		
	CHECKLAST.		
	/* Wait for response from client */		
	WAIT-FOR READ-RESPONSE OF hSocket.		
	CHECKLAST.		
	hsocket:READ).		
	CHECKLAST.		
	end.		

#89 - 12/24/2013 05:26 PM - Vadim Gindin

- File vig_upd20131225a.zip added

Here is an update, containing current solutions.

#90 - 01/06/2014 11:12 AM - Greg Shah

Constantin will be the one doing a more detailed review of your latest code.

I have taken a quick look at the code and only wanted to mention that there are some javadoc errors that should be cleaned up.

#91 - 01/07/2014 05:00 AM - Constantin Asofiei

The server project uses the LAST-EVENT:LABEL in only one place, with a case like (this is an assumption, as I can't find the sources corresponding to p1.p):

1. p1.p

```
wait-for close of current-window.
```

2. p2.p

```
run p1.p. /* press F4 here */
if last-event:label = "ENDKEY" then message "END-KEY pressed".
```

But this test produces a "PF4" label (on linux), not ENDKEY. Greg: can ENDKEY label (and last-key) be ever produced by user key press?

So, the plan is:

- make sure this case works properly
- merge with latest revision, cleanup the code (add javadoc, etc)
- add comments/TODOs in the code for all your concerns/cases which do not work properly
- get it regression tested.

#92 - 01/07/2014 08:41 AM - Greg Shah

Greg: can ENDKEY label (and last-key) be ever produced by user key press?

I'm not sure. Normally, I would have said "no" but there is this description in the 4GL reference:

```
For the LAST-EVENT handle, the LABEL attribute returns the names of low-level
events based on the EVENT-TYPE attribute value. For EVENT-TYPE = "KEYPRESS",
this attribute returns key label events, such as "F1" or "ESC". It also returns the key
labels of any keys that trigger key function events (returned by the FUNCTION
attribute).
```

The last part suggests that it may be possible.

Please send a note to Guy to see if he can provide more context about the use case and about what is in p1.p. Perhaps he can help us narrow down exactly what should be in the testcase to duplicate the condition.

#93 - 01/07/2014 09:20 AM - Constantin Asofiei

After contacting Guy, the conclusion is that the server project does not use the program which reads the LAST-EVENT:LABEL. But, we will still release the current implementation now. The plan at note 91 still applies, just ensure the basic tests work.

#94 - 01/07/2014 01:07 PM - Vadim Gindin

Constantin Asofiei wrote:

> .. just ensure the basic tests work.

I wrote in the table results of all tests I have. Do you mean, I should fix some of them? If you mean it, please specify what exactly.

#95 - 01/07/2014 01:08 PM - Vadim Gindin

I mean, what tests do you consider as basic?

#96 - 01/07/2014 01:47 PM - Constantin Asofiei

Vadim Gindin wrote:

> I mean, what tests do you consider as basic?

Case 8 from the table, widget and key events, plus what I've noted on note 91. If these work, follow the plan on note 91.

#97 - 01/08/2014 07:33 PM - Vadim Gindin

I tested your ENDKEY test.
Progress shows:
PF4 - END-ERROR [KEYPRESS]
Java shows:
F4 - END-ERROR [KEYPRESS]
Correct result. I'm going to run regression testing.

#98 - 01/09/2014 03:56 AM - Constantin Asofiei

> Correct result. I'm going to run regression testing.

Perfect. Post the update here, once you've added javadoc/todo's.

#99 - 01/09/2014 03:28 PM - Vadim Gindin

- File vig_upd20140109a.zip added

Here is the update. I'm waiting for your permission to start regression testing.

#100 - 01/09/2014 03:49 PM - Constantin Asofiei

Go ahead with the runtime testing. I'll do the review tomorrow.

#101 - 01/10/2014 10:03 AM - Constantin Asofiei

Vadim, I'm fine with the logic. All comments are about cleanup and javadoc/comment improvement:

- ClientState - you have a typo in the date's year (2012 instead of 2013)
- Keyboard - as for note 81 - why can't the initialization of DE_LOWEST and DE_HIGHEST be delayed? And eventCode doesn't use WorKArea, only keyCode does.
- Keyboard.isPGEvent - why are you formatting javadoc on 78 char lines? The max line length was increased to 98 some time ago... Also, always use an empty line between javadoc's "param" and "return" sections.
- LogicalTerminal
  - H228 should not be there (is for the window task).
  - even if currently some private methods are not added in the correct section, please group "public static" methods togheter. Now is after a "private static" method.
- ThinClient
  - functionKey - you have a " // same as last action code?" comment - is this still needed?
- KeyReader:
  - getLastFunction, getLabel: comments inside a method's body should be per line, not per group.
  - getLastFunction: the comment can be improved:

```
//      The FUNCTION attribute read at this step can orriginate from two locations:
//      - value from ClientState, cached in LogicalTerminal or
//      - value set explicitly through setLastKey() or readKey()
//
//      TODO: double check if this is read/written correctly, as it doesn't look right
```

  - setKeyFunctionWorker, setLabelWorker - check the javadoc, it has formatting issues; "link" should be used to reference something in javadoc
  - improve javadoc for WorkArea.eventType and keyFunction

#102 - 01/10/2014 04:29 PM - Vadim Gindin

Constantin Asofiei wrote:

..

- Keyboard - as for note 81 - why can't the initialization of DE_LOWEST and DE_HIGHEST be delayed? And eventCode doesn't use WorKArea, only keyCode does.

I tried and got an NPE.

- Keyboard.isPGEvent - why are you formatting javadoc on 78 char lines? The max line length was increased to 98 some time ago... Also, always use an empty line between javadoc's "param" and "return" sections.

Hmm.. I didn't know)

- LogicalTerminal
  - H228 should not be there (is for the window task).

I didn't understand. What do you mean here?

- ThinClient
  - functionKey - you have a " // same as last action code?" comment - is this still needed?

Yes, this comment is actual because the upper field actionCode have probably similar semantic.

The update had passed regression testing. Should I rerun it after these changes?

#103 - 01/13/2014 03:13 AM - Constantin Asofiei

Vadim Gindin wrote:

> Constantin Asofiei wrote:
> ..
>
> - Keyboard - as for note 81 - why can't the initialization of DE_LOWEST and DE_HIGHEST be delayed? And eventCode doesn't use WorKArea, only keyCode does.
>
> I tried and got an NPE.

Unless the DE_LOWEST and DE_HIGHEST are some codes from 4GL, is there a way to defer the initialization of these constants after WorkArea was created?

- H228 should not be there (is for the window task).

I didn't understand. What do you mean here?

This history entry has no place for this task, as the LT has no window-related changes (and the window-related update has not been released yet):

```
** 228 VIG 20131226          Added implementation of setCurrentWindow(..) and
**                           currentWindow(..) methods
```

- ThinClient
  - functionKey - you have a " // same as last action code?" comment - is this still needed?

Yes, this comment is actual because the upper field actionCode have probably similar semantic.

OK, then change it to something like: check if this field acts the same as last action code.

The update had passed regression testing. Should I rerun it after these changes?

If there are no logic changes, there is no need for testing again.

#104 - 01/13/2014 03:06 PM - Vadim Gindin

Constantin Asofiei wrote:

> ..
> Unless the DE_LOWEST and DE_HIGHEST are some codes from 4GL, is there a way to defer the initialization of these constants after WorkArea was created?

These codes (551-560) are document Progress key codes. As I understand they cannot be changed. Only key functions can be changed through environment (PROTERMCAP file for Unix) - not key codes. So there are following variants:
1. Stay these constants as numbers (current).
2. Calculate them during each execution of Keyboard.isDevEvent() method in following way:

```
public static boolean isDevEvent(int lastkey)
{
   return (lastkey >= keyCode("U1") && lastkey <= keyCode("U10"));
}
```

3. Make new 2 fields for them in WorkArea and calculate them in constructor.

I would prefer the current implementation (n1) because of its simplicity.

#105 - 01/13/2014 03:17 PM - Constantin Asofiei

Ok, then stay with option 1 (constants) but make sure to document this in the javadoc for the fields (i.e. they are progress constants too).

#106 - 01/13/2014 04:33 PM - Vadim Gindin

committed to bzr rev #10435

#107 - 01/13/2014 04:53 PM - Greg Shah

Please post the final update zip here and we will close the task.

#108 - 01/13/2014 04:54 PM - Vadim Gindin

- File vig_upd20140109a.zip added

Here it is. Implementation of basic behavior

#109 - 01/13/2014 05:12 PM - Greg Shah

- Status changed from New to Closed

#110 - 01/14/2014 03:00 AM - Vadim Gindin

- File vig_upd20140109a.zip added

Error fix. Committed to bzr, rev #10436.

#111 - 01/14/2014 08:57 AM - Constantin Asofiei

- File deleted (vig_upd20140109a.zip)

#112 - 01/14/2014 08:58 AM - Constantin Asofiei

- File vig_upd20140114a.zip added

Attached update contains the correct changes applied to rev 10436 (compared to 10434).

#113 - 07/25/2015 08:55 AM - Greg Shah

LAST-EVENT:EVENT-TYPE = "MOUSE" needs to be supported for GUI

#114 - 07/30/2015 04:03 PM - Constantin Asofiei

Greg Shah wrote:

> LAST-EVENT:EVENT-TYPE = "MOUSE" needs to be supported for GUI

This is already added: KeyReader.setLastKey sets it to MOUSE_EVTTYPE if the client-side has sent a mouse virtual key (via StateSynchronizer) to the server-side. Have you found a use-case which is not working?

#115 - 07/30/2015 04:42 PM - Greg Shah

> Have you found a use-case which is not working?

No. Sometime in the past I had written down a note to myself that this was still needed. I guess it was added after that and I didn't properly track that fact.

I'm going to close this task. Is there any reason I should not?

#116 - 03/23/2016 01:03 PM - Greg Shah

Can I close #2226 or is there still some work to do on LAST-EVENT?

#117 - 03/23/2016 02:01 PM - Constantin Asofiei

Greg Shah wrote:

> Can I close #2226 or is there still some work to do on LAST-EVENT?

These attributes I don't think are implemented for LAST-EVENT:

```
COLUMN attribute
ROW attribute
INSTANTIATING-PROCEDURE attribute
ON-FRAME-BORDER attribute
```

#118 - 03/23/2016 02:37 PM - Greg Shah

Also, the LAST-EVENT:X and LAST-EVENT:Y are only stubbed in the runtime (see KeyReader.getEventY() and KeyReader.getEventX(). Unfortunately, these are actually used in 1 place in the current customer GUI app.

#119 - 03/27/2016 10:26 AM - Constantin Asofiei

Created task branch 2226a from trunk rev 10986.

Revision 10987 adds (Hynek, please do a review):

1. conversion of X/Y attributes for both LAST-EVENT and widgets
2. runtime support for LAST-EVENT:X/Y attributes. The rules basically are:
   - the coordinate is relative to top-left corner for the viewport of frame or window workspace.
   - window's border, status area, message area or title do not raise portable mouse events
   - frame title does not raise portable mouse events
   - scroll bar and scroll buttons do not raise portable mouse events
   - when hovering a LABEL instance, the "mouse source" is actually the parent frame, not the LABEL
   - a disabled widget can not be returned by findMouseSource - if it is disabled, the parent FRAME is the source
   - overlay windows (like combo drop-down or ALERT-BOX) do not raise portable mouse events
3. fix for FRAME:TYPE in case of DIALOG-BOX frames
4. fixed FRAME:X/Y attribute setter/getter
5. fixed (de)registration of LABEL instances (side or header labels), plus delimiter, in case of FRAMEs (this fixes the "zombie mouse source" messages)
6. I've moved some code from 1811t here (related to window border drawing, in revision 1811t rev 10920)

What is left:

1. testing with scrollable widgets: FRAME, WINDOW, EDITOR, SELECTION-LIST, COMBO-BOX in simple mode
2. testing with MENU and POPUP-MENU
3. testing with other portable mouse events (I've tested with mouse-click only until now - mouse move is mostly of interest in case of scrollbars/popupmenus/etc)

Other issues:

1. the viewport (and scrollable area) height for a FRAME with a TITLE is not computed correct - it includes the title height, thus the bottom border is not drawn correctly
2. combo-box drop-down (with trunk) can't be opened - it flashes on screen; see bellow for test
3. dialog-box layout is incorrect; see bellow for test
4. I can't make a FRAME scrollable, if its given width is less than the widget's width - 4GL shows scrollbars, P2J doesn't

The test:

```
def var ch as char.
def var cb as char view-as combo-box list-items "a", "b", "c", "d".
form ch cb with frame f1.
update ch cb with frame f1.

form ch cb with frame f3 title "aa" view-as dialog-box side-labels.
update ch cb with frame f3.
```

#120 - 03/28/2016 07:08 AM - Hynek Cihlar

Constantin Asofiei wrote:

> Created task branch 2226a from trunk rev 10986.
>
> Revision 10987 adds (Hynek, please do a review):

Just couple of minor things.

In KeyReader.setY() readOnlyError("X") --> readOnlyError("Y").

In GuiWindowCommons.insideBorder() and GuiWIndowCommons.drawBorder() you construct the individual border rectangles one pixel bigger. Either use the ctor NativeRectangle(NativePoint, NativeDimension) or subtract one from width and height passed to NativeRectangle(top, left, bottom, right).

#121 - 03/28/2016 08:03 AM - Constantin Asofiei

Hynek Cihlar wrote:

> Constantin Asofiei wrote:
>
> > Created task branch 2226a from trunk rev 10986.
> >
> > Revision 10987 adds (Hynek, please do a review):
>
> Just couple of minor things.

Thanks, I'll fix them.

Can you give me an working example of scrollable frame, in P2J? Also, in getPortableMouseEventCoordinates() I need to consider the viewport's scrolled origin: i.e. if the viewport was scrolled 10 pixels to the right/bottom, then I need to substract 10 pixels from x/y coordinates. Can you give me an API which determines the scrolled amount, in pixels?

#122 - 03/28/2016 08:23 AM - Hynek Cihlar

Constantin Asofiei wrote:

> Hynek Cihlar wrote:
>
> > Constantin Asofiei wrote:
> >
> > > Created task branch 2226a from trunk rev 10986.
> > >
> > > Revision 10987 adds (Hynek, please do a review):
> >
> > Just couple of minor things.
>
> Thanks, I'll fix them.
>
> Can you give me an working example of scrollable frame, in P2J? Also, in getPortableMouseEventCoordinates() I need to consider the viewport's scrolled origin: i.e. if the viewport was scrolled 10 pixels to the right/bottom, then I need to substract 10 pixels from x/y coordinates. Can you give me an API which determines the scrolled amount, in pixels?

Scrollable frame example:

```
DEF FRAME f WITH SIZE 30 BY 10.
DEF VAR cb AS CHAR FORMAT "x(3)" VIEW-AS COMBO-BOX  LIST-ITEMS
   "item1", "item2", "item3", "item4", "item5" SIZE 10 BY 5.
FRAME f:VIRTUAL-WIDTH = 100.
FRAME f:VIRTUAL-HEIGHT = 100.
ENABLE cb WITH FRAME f SIDE-LABELS.
WAIT-FOR GO OF FRAME f.
```

I don't think we have an API for reading the scrolled offset of the frame content. To get the offset call physicalLocation() on the frame's SensitiveScrollContainer (Frame.contentPane).

#123 - 03/28/2016 08:25 AM - Hynek Cihlar

Constantin Asofiei wrote:

1. I can't make a FRAME scrollable, if its given width is less than the widget's width - 4GL shows

Please post the example. I just also found a case where the frame doesn't show scroll when it should. I will create a new Redmine issue and put the cases there.

#124 - 03/28/2016 08:55 AM - Constantin Asofiei

Hynek Cihlar wrote:

> Constantin Asofiei wrote:
>
> 1. I can't make a FRAME scrollable, if its given width is less than the widget's width - 4GL shows
>
> Please post the example. I just also found a case where the frame doesn't show scroll when it should. I will create a new Redmine issue and put the cases there.

See testcases/uast/lastevent/lastevent_mouse_coords.p - the inner frame f2 must be scrollable (with its size set via the SIZE frame option).

#125 - 03/28/2016 08:55 AM - Constantin Asofiei

Hynek Cihlar wrote:

> I don't think we have an API for reading the scrolled offset of the frame content. To get the offset call physicalLocation() on the frame's SensitiveScrollContainer (Frame.contentPane).

I need something similar for window's workspace - do you have any hints?

#126 - 03/28/2016 09:51 AM - Constantin Asofiei

2226a rev 10988 adds the remainder functionalities (exclude scrollbars and adjust for content's scrolled location).

#127 - 03/28/2016 10:00 AM - Greg Shah

Thank you for providing the good detail in note 119. Implicit in that note was the idea that if no portable mouse events are generated, then the LAST-EVENT:X/Y is not updated. In other words, low-level mouse events don't change the LAST-EVENT:X/Y. Do I understand correctly?

#128 - 03/28/2016 10:20 AM - Hynek Cihlar

Constantin Asofiei wrote:

> Hynek Cihlar wrote:
>
> > I don't think we have an API for reading the scrolled offset of the frame content. To get the offset call physicalLocation() on the frame's SensitiveScrollContainer (Frame.contentPane).
>
> I need something similar for window's workspace - do you have any hints?

WindowWorkspace also uses NativeScrollContainer so similar approach as for Frame,
WindowWorkspace.viewport().getScrollWidget().physicalLocation() will give you the scroll offset.

#129 - 03/28/2016 10:22 AM - Constantin Asofiei

Greg Shah wrote:

> Thank you for providing the good detail in note 119. Implicit in that note was the idea that if no portable mouse events are generated, then the LAST-EVENT:X/Y is not updated. In other words, low-level mouse events don't change the LAST-EVENT:X/Y. Do I understand correctly?

Portable mouse events and low level mouse events (i.e. mouse-select-click and left-mouse-click) behave the same. The idea is that events other than mouse events do not update the LAST-EVENT:X/Y attributes.

And, there is another issues to test/fix; the LAST-EVENT:X/Y attrs are not "scoped": if there are nested event processing loops, I think these attribute get "reset" (set to unknown) when the event processing loop starts. Current support in 2226a is "scoped", which is not correct.

#130 - 03/29/2016 06:04 AM - Constantin Asofiei

2226a rev 10989 fixes this: LAST-EVENT:X/Y are reset at the beginning of an event processing loop (readkey, pause, wait-for, etc) - these attrs are not scoped

Next step is runtime testing, I can't think of anything else to check.

#131 - 03/29/2016 09:58 AM - Greg Shah

Code Review Task Branch 2226a Revision 10989

Everything is fine.

I don't understand this code in ThinClient.processProgressEvent():

```
            boolean useMouseSource = mouseSource.config() != null &&
                                     (!WidgetId.virtualWidget(mouseSource.config().id) ||
                                      mouseSource instanceof Label);
```

Labels now are in the registry and have a valid (non-negative) ID, so I would have expected this code to not have to explicitly reference Label, unless it was trying to exclude it like this !(mouseSource instanceof Label).

#132 - 03/29/2016 10:13 AM - Constantin Asofiei

Greg Shah wrote:

> Code Review Task Branch 2226a Revision 10989
>
> Everything is fine.
>
> I don't understand this code in ThinClient.processProgressEvent():
>
> [...]
>
> Labels now are in the registry and have a valid (non-negative) ID, so I would have expected this code to not have to explicitly reference Label, unless it was trying to exclude it like this !(mouseSource instanceof Label).

Almost every widget (virtual or not) is added to the registry (I think there are some exceptions, when the ID is actually null - like for the Viewport). But, currently labels don't have a server-side counterpart (thus are still "virtual" in P2J's point-of-view), and they have a negative ID. This code tells P2J to use the label as a mouse source, and not the window.

#133 - 03/29/2016 10:21 AM - Greg Shah

OK, please go ahead with runtime testing.

#134 - 03/29/2016 01:18 PM - Constantin Asofiei

Greg Shah wrote:

> OK, please go ahead with runtime testing.

Runtime testing was started.

#135 - 03/30/2016 06:22 AM - Hynek Cihlar

Constantin Asofiei wrote:

> Hynek Cihlar wrote:
>
> > Please post the example. I just also found a case where the frame doesn't show scroll when it should. I will create a new Redmine issue and put the cases there.
>
> See testcases/uast/lastevent/lastevent_mouse_coords.p - the inner frame f2 must be scrollable (with its size set via the SIZE frame option).

I've added the scrolling bug to #3015 note 2.

#136 - 03/30/2016 06:51 AM - Constantin Asofiei

Hynek Cihlar wrote:

> Constantin Asofiei wrote:
>
> > Hynek Cihlar wrote:
> >
> > > Please post the example. I just also found a case where the frame doesn't show scroll when it should. I will create a new Redmine issue and put the cases there.
> >
> > See testcases/uast/lastevent/lastevent_mouse_coords.p - the inner frame f2 must be scrollable (with its size set via the SIZE frame option).

> I've added the scrolling bug to #3015 note 2.

Another issue: after 2226a will be released, you will be able to use mouse events to see the location of i.e. a click event, in the scrolled frame. With scrollable frames, you will see that there are two other issues:

1. when scrolling, the scrollbar's thumb size increases when i.e. scrolled down
2. the scrolled amount doesn't match the frame's scrollable size: I've reached a point reported as 1000px on the Y axis...
3. the widgets are not drawn partially, when scrolled

All these can be seen in the lastevent/lastevent_mouse_coords.p test.

#137 - 03/30/2016 07:18 AM - Hynek Cihlar

Constantin Asofiei wrote:

> Hynek Cihlar wrote:
>
> > Constantin Asofiei wrote:
> >
> > > Hynek Cihlar wrote:
> > >
> > > > Please post the example. I just also found a case where the frame doesn't show scroll when it should. I will create a new Redmine issue and put the cases there.
> > >
> > > See testcases/uast/lastevent/lastevent_mouse_coords.p - the inner frame f2 must be scrollable (with its size set via the SIZE frame option).
> >
> > I've added the scrolling bug to #3015 note 2.
>
> Another issue: after 2226a will be released, you will be able to use mouse events to see the location of i.e. a click event, in the scrolled frame. With scrollable frames, you will see that there are two other issues:
>
> 1. when scrolling, the scrollbar's thumb size increases when i.e. scrolled down
> 2. the scrolled amount doesn't match the frame's scrollable size: I've reached a point reported as 1000px on the Y axis...
> 3. the widgets are not drawn partially, when scrolled
>
> All these can be seen in the lastevent/lastevent_mouse_coords.p test.

Ok, I've copied the note to #3015.

#138 - 03/30/2016 09:42 AM - Constantin Asofiei

2226a runtime testing passed.

#139 - 03/30/2016 10:42 AM - Constantin Asofiei

Rebased 2226a from trunk rev 10988 - new 2226a rev 10991.

#140 - 03/30/2016 11:54 AM - Greg Shah

Please merge 2226a to trunk.

#141 - 03/30/2016 12:03 PM - Constantin Asofiei

Greg Shah wrote:

> Please merge 2226a to trunk.

2226a was merged to trunk rev 10989 and archived.

#142 - 11/16/2016 11:42 AM - Greg Shah

- Target version changed from Milestone 7 to Runtime Support for Server Features

Files

ca_upd20130223b.zip	15.7 KB	02/23/2013	Constantin Asofiei
ca_upd20130223c.zip	15.8 KB	02/23/2013	Constantin Asofiei
ca_upd20130223d.zip	16.1 KB	02/23/2013	Constantin Asofiei
client_master_2813.log	3.01 KB	11/12/2013	Vadim Gindin
server-socket-error.log	6.38 KB	11/12/2013	Vadim Gindin
vig_upd20131115a.zip	36.2 KB	11/15/2013	Vadim Gindin
vig_upd20131118a.zip	37.4 KB	11/18/2013	Vadim Gindin
vig_upd20131122a.zip	52.3 KB	11/21/2013	Vadim Gindin
vig_upd20131125a.zip	56.6 KB	11/25/2013	Vadim Gindin
vig_upd20131129a.zip	176 KB	11/28/2013	Vadim Gindin
vig_upd20131211a.zip	226 KB	12/11/2013	Vadim Gindin
vig_upd20131225a.zip	227 KB	12/24/2013	Vadim Gindin
vig_upd20140109a.zip	215 KB	01/09/2014	Vadim Gindin
vig_upd20140109a.zip	229 KB	01/13/2014	Vadim Gindin
vig_upd20140114a.zip	230 KB	01/14/2014	Constantin Asofiei