User Interface - Bug #2497

ANY-PRINTABLE only works for specific widget triggers, but not frame or global anywhere

01/22/2015 05:48 PM - Greg Shah

Status:	Closed	**Start date:**	01/22/2015
Priority:	Normal	**Due date:**	
Assignee:	Greg Shah	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:	GUI Support for a Complex ADM2 App		
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 01/22/2015 05:49 PM - Greg Shah

The following testcase illustrates this problem:

```
def var trig-scope as int init 1
   label "Trigger Scope"
   view-as radio-set radio-buttons "Widget", 1, "Frame", 2, "Anywhere", 3.

update trig-scope with frame ts-fr.

def var txt1 as char.
def var txt2 as char.
def var other as char.

def frame f-one txt1 txt2.
def frame f-two other.

case trig-scope:
   when 1 then
   do:
      on any-printable of txt1
      do:
         message "WIDGET ANY-PRINTABLE".
         pause.
         hide message.
      end.
   end.

   when 2 then
   do:
      on any-printable of frame f-one anywhere
      do:
         message "FRAME ANY-PRINTABLE".
         pause.
         hide message.
      end.
   end.

   when 3 then
   do:
      on any-printable anywhere
      do:
         message "ANYWHERE ANY-PRINTABLE".
         pause.
         hide message.
      end.
   end.
end.

/* use ESC-TAB to switch focus between frames */
enable all with frame f-one.
```

```
enable all with frame f-two.

wait-for go of frame f-one.
```

Case 1 works fine on P2J, but all 3 cases work in the 4GL.

#2 - 01/22/2015 05:53 PM - Greg Shah

- File ges_upd20150122a.zip added

This is the fix. It ensures that ANY-PRINTABLE and ANY-KEY are searched on each pass through (specific-widget, frame-wide, global anywhere).

Constantin: please review it. I'm sorry this one will be harder to review because it conflicts with your change in #2495. I know that I will have to merge with yours (I want your change to be checked in first).

#3 - 01/22/2015 05:55 PM - Greg Shah

The testcase is checked in as testcases/uast/any_printable_event.p.

#4 - 01/23/2015 01:45 PM - Greg Shah

- File ges_upd20150123a.zip added

Merged update to bzr 10719 which includes the fix from #2495. I'm going to put this into testing.

#5 - 01/23/2015 03:03 PM - Constantin Asofiei

Greg Shah wrote:

> Merged update to bzr 10719 which includes the fix from #2495. I'm going to put this into testing.

I can't see anything wrong with the logic... testing will tell us more.

Note that the javadoc for EventList.lookupEventHelper method is missing.

#6 - 01/23/2015 03:45 PM - Greg Shah

Good catch. Here is the version with javadoc.

#7 - 01/23/2015 03:46 PM - Greg Shah

- File ges_upd20150123b.zip added

No, here it is.

#8 - 01/24/2015 02:53 PM - Greg Shah

- Status changed from WIP to Closed

Passed regression testing and checked in as bzr rev 10720.

#9 - 01/28/2015 04:03 PM - Greg Shah

This update (which does make ANY-PRINTABLE more widely functional), can also cause problems. This code that applies the resulting modified key event that originally matched the ANY-PRINTABLE event is broken:

```
def var txt as char.

def frame f-edit txt format "X(2)".
def var i as int init 1.

on any-printable of frame f-edit anywhere
do:
   hide message.
   message i.
   i = i + 1.
   apply if lastkey >= 65 and lastkey <= 255 then asc(caps(chr(lastkey))) else lastkey.
   return no-apply.
end.

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

message txt:screen-value.
```

In the 4GL, the ANY-PRINTABLE trigger is only executed once. In P2J, we execute the trigger twice and then detect an error and force the cursor position to the beginning of the field.

I'm looking for a way to make the ANY-PRINTABLE avoid recursion. Strangely, we don't seem to have any general purpose recursion avoidance approach in the client. I wonder if this problem is more general.

#10 - 01/28/2015 07:11 PM - Greg Shah

Actually, we do have generic protection logic for trigger recursion. It is located in LogicalTerminal.trigger():

```
      // avoid recursion by looking up the widget in the trigger stack
```

```
      // and the event in the synced event stack and bypass it if the pair
      // matches

      for (int i = 0; i < lt.triggerResource.size(); i ++)
      {
         if (!lt.triggerResource.get(i).equals(res))
         {
            continue;
         }

         // recursion of the widget; check if the recursion for the event
         if (lt.triggerEvent.get(i).intValue() == eventId)
         {
            // the pair matches entirely; ignore trigger
            return trv;
         }
      }

      // call the trigger
      lt.triggerResource.addFirst(res);
      lt.triggerEvent.addFirst(new Integer(eventId));
```

The problem is that the ANY-KEY or ANY-PRINTABLE are only considered for during the lookup to find if there is a matching trigger. When the trigger is invoked, we don't send any notification that the event has been "overridden" instead the original event is sent. So in this case we can't find the combination of event and widget because: if the first call is with a lowercase character, that character is what is in the list, not the ANY-PRINTABLE event. Then the trigger body uppercases the character and applies it. The ANY-PRINTABLE matches again but when we get to the server the event is the uppercased character so it doesn't match the lowercase character in the event list. So a lowercase character will execute this trigger twice but an uppercase character will detect the recursion and return without executing the trigger.

I'm working on a solution.

#11 - 01/29/2015 01:02 PM - Greg Shah

- File ges_upd20150129b.zip added

This update fixes the trigger recursion but the cursor is still always forced back to the first char position. I have debugged through the code extensively and I cannot find the location where we go wrong.

One thing I tried was to bypass (manually in the debugger) the 2nd trigger execution attempt to see if the extra trip to the server and the

accompanying state changes in the client were causing the problem. They don't appear to have any affect, so I didn't move the recursion protection from LogicalTerminal to ThinClient, though eventually we probably should do that.

Part of the problem is that the screen refreshes are deferred so I can't see when the cursor moves until the screen is repainted. But I've checked the most obvious places and can't find anything wrong.

Any ideas are much appreciated.

#12 - 01/29/2015 01:25 PM - Constantin Asofiei

Greg Shah wrote:

> Part of the problem is that the screen refreshes are deferred so I can't see when the cursor moves until the screen is repainted. But I've checked the most obvious places and can't find anything wrong.
>
> Any ideas are much appreciated.

The problem is that after APPLY is executed in the test in note 9 (the trigger is ignored, as the server-side does not allow it), P2J sends a FOCUS-GAINED event to the widget. Place a breakpoint in TC.apply:3023:

```
         tk.setInvalidate(true);
         applyWorker(event);
         tk.setInvalidate(event.source(), false, true);
```

```
         boolean targetValid = inFocus instanceof FixedSizeContainer &&
                               inFocus.isVisible()             &&
                               inFocus.isEnabled()             &&
                               (tmpFrame == null ||
                                tmpFrame.isVisible());
         if (triggerNesting > 0 && !frameChangeEvent)
         {
            if (!explicit)
            {
               refocusTarget(widgetId, inFocus, targetValid); // this line here
            }
```

The refocusTarget call will re-focus the widget, which in turn will place the cursor on the first char.

I guess we need a way of knowing if the APPLY was ignored or not?

#13 - 01/29/2015 02:26 PM - Greg Shah

I stepped over that refocusTarget() call because I knew the target widget was already in focus.

I don't fully understand the way we've coded this. The explicit == false call comes from the server in the LT.apply(long). This version is meant to apply the event to the currently focused widget. It seems that the widgetId will always be the same as the currently focused widget, so long as there is a widget in focus. So if the widgetId is not -1, then it should always already be in focus, or else we have some kind of state problem for the LT.focus() to report the wrong widget.

It seems like we could use FocusManager.tryFocusChange() instead of the unconditional setFocusOn(). Doing so would ensure that no bogus focus change occurs, but I don't know what scenarios are dependent upon the setFocusOn(). This could break things badly.

#14 - 01/29/2015 02:43 PM - Greg Shah

The change to add refocusTarget() came in rev 9907 (H649 on 20101015 by SIY). There is no indication of why this was done.

#15 - 01/29/2015 03:26 PM - Constantin Asofiei

@refocusTarget@ might have been used to solve a case like this:

```
def var ch1 as char.
def var ch2 as char.
def var i as int.

form ch1  ch2 with frame f-edit.

on "2" anywhere do:
   message "2 before " focus:name.
   apply "entry" to ch2 in frame f-edit.
   message "2 after " focus:name.
end.

on "1" anywhere do:
   message "1 before " focus:name.
   apply "2".
   message "1 after " focus:name.
end.

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

Execute the test this way:

1. input "345" in ch1

```
┌──────────────────┐
│ch1      ch2      │
│──────── ──────── │
│345               │
└──────────────────┘
```

2. press "1" - it will show you how focus is switched when "2" trigger is invoked.

```
┌──────────────────┐
│ch1      ch2      │
│──────── ──────── │
│345               │
└──────────────────┘
...
1 before  ch1
2 before  ch1
```

```
Press space bar to continue.
```

3. after apply "2" ends, focus is again on ch1

```
┌──────────────────┐
│ch1      ch2      │
│──────── ──────── │
│345               │
└──────────────────┘
...
2 after  ch2
1 after  ch1
Enter data or press F4 to end.
```

Commenting the refocusTarget call gives some weird results: ch1 ends up with the "21" text after "1" is pressed, instead of the cursor just moving to the first char... With refocusTarget on, the result is still wrong, but the ch1 shows only "1".

#16 - 01/29/2015 03:37 PM - Greg Shah

That is a strange result. I wouldn't have expected that without NO-APPLY. The entry/leave processing must be responsible for bypassing the default processing of the fill-in in the 4GL.

#17 - 01/29/2015 03:54 PM - Constantin Asofiei

Greg Shah wrote:

> That is a strange result. I wouldn't have expected that without NO-APPLY. The entry/leave processing must be responsible for bypassing the default processing of the fill-in in the 4GL.

Something else strange is that, in the screen at step 3 in note 15, regardless of the printable key you press, it will not be applied to ch1. If you press left-arrow, you will see the caret positioned at the end of the ch1 fill-in... I guess the caret is "out-of-the-editable-area" and the key can't be applied to the widget.

Anyway, back to my original thought why I was building this test: I was trying to determine a case when an implicit APPLY (called from a trigger) can switch focus to something other than the trigger's focus, and see how it gets restored after the APPLY finishes.

#18 - 01/29/2015 03:57 PM - Greg Shah

I've tried the following idea:

```
            if (!explicit)
            {
               if (inFocus.config().id.getId() != widgetId)
               {
                  refocusTarget(widgetId, inFocus, targetValid);
               }
            }
            else
```

It fixes the problem with ANY-PRINTABLE, but it breaks your case (it is the same result as commenting out the refocusTarget().

Then I tried this:

```
            if (!explicit)
            {
               Widget currentFocus = UiUtils.getCurrentFocus();

               if (currentFocus == null || currentFocus.config().id.getId() != widgetId)
               {
                  refocusTarget(widgetId, inFocus, targetValid);
               }
            }
            else
```

This gives the same results (works for my case, breaks yours). Even with refocusTarget() unconditionally executed, P2J doesn't bypass the default event processing. The flow of the event processing and the currently in-focus widget is correct with both of my approaches and with the unconditional refocusTarget().

I wonder if it is worth testing my second approach with MAJIC.

#19 - 01/29/2015 04:09 PM - Constantin Asofiei

Greg Shah wrote:

> I wonder if it is worth testing my second approach with MAJIC.

Yes, I think this one is OK to test.

On a side note, there are the testcases/uast/*focus*.p tests... maybe one of them covers the original cause why refocusTarget was introduced.

#20 - 01/29/2015 04:11 PM - Greg Shah

> there are the testcases/uast/*focus*.p tests.

Good idea. I'll see if they can be used to test this out.

#21 - 01/30/2015 06:56 AM - Greg Shah

- File ges_upd20150130b.zip added

- File hc_upd20150130b.zip added

The changes are doing well in regression testing. I got timeout failures in tc_item_master_006, tc_item_master_104 and tc_gl_003 which look like system load issues. I'm re-running main testing right now.

I have not had the time to run all the standalone focus tests. I'm not sure I'll get to it.

Hynek's layout changes are going to be checked in first. The latest version is attached and my latest version is also attached. Mine is merged on top of his.

#22 - 01/30/2015 08:12 AM - Constantin Asofiei

I'm noting this here, but is a bug, not a regression; the following case doesn't work in P2J:

```
def button btny label "yes".
def button btnn label "no".

def var ch as char.

form ch with frame f1 size 80 by 20.

on "f4", "pf4" of frame f1 do:
```

```
   form "" with frame ff2 size 30 by 15 row 7 centered no-labels overlay.
   form btny btnn with  no-box frame ff overlay row 8 size 28 by 13 centered.

   pause 0 before-hide.
   view frame ff2.

   update btny btnn with frame ff.
end.

update ch with frame f1.
```

F4 is not picked up by P2J.

#23 - 01/30/2015 08:16 AM - Constantin Asofiei

LE: the trigger in note 22 was changed to be for the frame, not widget.

#24 - 11/16/2016 12:13 PM - Greg Shah

- Target version changed from Milestone 12 to GUI Support for a Complex ADM2 App

Files

ges_upd20150122a.zip	14.2 KB	01/22/2015	Greg Shah
ges_upd20150123a.zip	14.4 KB	01/23/2015	Greg Shah
ges_upd20150123b.zip	14.7 KB	01/23/2015	Greg Shah
ges_upd20150129b.zip	154 KB	01/29/2015	Greg Shah
hc_upd20150130b.zip	325 KB	01/30/2015	Greg Shah
ges_upd20150130b.zip	154 KB	01/30/2015	Greg Shah