User Interface - Bug #2365

ThinClient.trigger() and ThinClient.validate() must send the screen-buffer for all frames which are in input mode, not just the one on which the event was fired

08/07/2014 10:14 AM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Eugenie Lyzenko	**% Done:**	100%
Category:		**Estimated time:**	4.00 hours
Target version:	Cleanup and Stabilization for GUI		
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

History

#1 - 08/07/2014 10:16 AM - Greg Shah

The original discussion in which this issue was identified:

-------- Original Message --------
Subject: Re: GUI SCREEN-VALUE (and Character INPUT <keyword)
Date: Thu, 07 Aug 2014 09:59:18 -0400
From: Greg Shah
To: <customer>
CC: ...

<customer>,

> There are no triggers in proc1.p - so I'm not sure when or how from your description the input data from frame1 would get passed to the P2J server. Is it on every keystroke, or just when the user hits the OK button, or on another event (e.g. leave of the fill-in or frame)?

This scenario looks like a hole in our current implementation. Currently, we send the screen buffer associated with the frame upon which the trigger was fired. The proper approach would be to send all screen buffers which are in "input" mode (enabled).

We will fix that.

> Following what you've written, if once you're in the P2J server code, you only make use of the screen buffers held on the P2J server side, then that information must either have already been passed across, or be passed across as part of the call from the P2J client.

Yes.

Thanks,
Greg

On 08/07/2014 04:08 AM, <customer> wrote:

> Hi Greg,
> Thanks for the reply, I'm still a bit confused though about the passing of the screen buffers.
> Take the following example
> proc1.p
>
> defines frame1
> frame1 contains a single fill-in widget, with no triggers.
> has an internal procedure that returns the INPUT-VALUE of the fill-in.
>
> proc2.p
>
> defines frame2

frame2 contains a fill-in and a button.
There's a trigger on the button which calls the internal procedure in proc1.p

Both proc1.p and proc2.p are run persistently, and are displayed and enabled.
There are no triggers in proc1.p - so I'm not sure when or how from your description the input data from frame1 would get passed to the P2J server. Is it on every keystroke, or just when the user hits the OK button, or on another event (e.g. leave of the fill-in or frame)?
When the trigger fires on the button in proc2.p, will it only pass the screen buffer associated with frame2, or does it pass the screen buffers associated with every enabled or displayed or updated frame?
Following what you've written, if once you're in the P2J server code, you only make use of the screen buffers held on the P2J server side, then that information must either have already been passed across, or be passed across as part of the call from the P2J client.
I hope you get the gist of what I'm asking.
Thanks,
<customer>

On 6 August 2014 17:08, Greg Shah <ges@goldencode.com> wrote:

<customer>,

> the server can call back to the java client whenever it hits a SCREEN-VALUE or INPUT-VALUE keyword, to get the current values, and you are also storing those values in the P2J server as well.

Not quite.

There are 4 possible directions that control flow can transfer:

1. server calls client (to invoke some client-only feature on behalf of the business logic)
2. client returns to server (from processing some requested client-only feature)
3. client calls server (to request processing of a trigger or validation expression)
4. server returns to client (from processing a trigger or validation expression)

In each one of these control flow transfers, the latest screen buffer may be sent to the other side as part of the transfer. Only those features which could possibly use a screen buffer, will cause it to be transferred, but that is just an optimization. The key point is that since the converted application processing can only ever be running on the server or the client (but never both at once), so long as we properly transfer the latest screen buffer state as part of these 4 control flow transfers, then both sides will always "see" the latest screen buffer when needed.

Because of this synchronized data structure, when the converted application encounters a SCREEN-VALUE, INPUT-VALUE, FRAME-VALUE or INPUT it just needs to ask the server-side frame for the updated value. The frame consults its screen buffer and reports the result. There is no need to call down to the client to get it, the latest values are already there.

Since converted 4GL code doesn't run on the client, there is no such thing as SCREEN-VALUE, INPUT-VALUE, FRAME-VALUE or INPUT on the client side. Nevertheless, where the screen buffer is accessed internally to implement such things as DISPLAY or SET, the latest screen buffer is there and any changes will be sent back to the server on the next control flow transfer.

> Sort of related to this, with triggers, do you put listeners on the widgets in the java client, only when a trigger is required? Or do you send every user event (keystroke, mouse button press etc...) up to the P2J layer?

When we call down to the client to invoke some operation that is considered an input operation (i.e. the user will be feeding keys into the application), the parameters for that remote call will include a list of trigger definitions.

The client-side event processing scans this list for matches on every keystroke. There is no need to go to the server with every keystroke (except for code like EDITING blocks that are explicitly designed as a READKEY loop). Instead, we only invoke triggers on the server side when we know there is a match.

This is not strictly be implemented as the "listeners" pattern, but the idea is close enough.

By the way, to us the "P2J layer" exists at both the server and client, so we more properly say "P2J server" or "P2J client" when discussing control flow transfers.

> On the other hand, if you do put listeners on the widgets on an as needed basis, then we still have some fill-in's which have triggers on every keystroke, - e.g. to put the entry into upper case. We do this by capturing the events, and only displaying the screen value in caps.
> E.g. there are many instances of code such as this:
> ON ANY-PRINTABLE OF RowObject.agifcinterface_code IN FRAME F-Main /* Interface Code */
> DO:
> APPLY KEYCODE)) TO {&SELF-NAME}.
> RETURN NO-APPLY.
> END.
> So I guess a roundtrip from the webclient to the P2J server will be required to render any such character.

Yes.

> Are you going to push the listeners down from the java client to the webclient, or will you have to send every event from the webclient to the java client to before characters can be rendered?

I do think it is likely that we will provide the short list of events for which we need to notify the client. This is made more tricky by the high level function/event processing like CHOOSE, which is something that is implemented in common client code today, but which we may need to do something more advanced to allow the web client to detect this itself.

We simply don't yet know exactly how this will work. This is part of the decision on "how we will split the client functionality between Java and web" which I was mentioning in the meeting.

Thanks,
Greg

On 08/06/2014 11:21 AM, <customer> wrote:

> Hi Greg,
> Thanks for your detailed reply, I think I follow what you're saying: that the server can call back to the java client whenever it hits a SCREEN-VALUE or INPUT-VALUE keyword, to get the current values, and you are also storing those values in the P2J server as well. Is that correct?
> Sort of related to this, with triggers, do you put listeners on the widgets in the java client, only when a trigger is required? Or do you send every user event (keystroke, mouse button press etc...) up to the P2J layer?
> If it's every keystroke, then I guess, you can't display characters until the P2J server has passed back that it is OK to display characters?
> On the other hand, if you do put listeners on the widgets on an as needed basis, then we still have some fill-in's which have triggers on every keystroke, - e.g. to put the entry into upper case. We do this by capturing the events, and only displaying the screen value in caps.
> E.g. there are many instances of code such as this:
> ON ANY-PRINTABLE OF RowObject.agifcinterface_code IN FRAME F-Main /* Interface Code */
> DO:
> APPLY KEYCODE)) TO {&SELF-NAME}.
> RETURN NO-APPLY.
> END.
> So I guess a roundtrip from the webclient to the P2J server will be required to render any such character.
> Are you going to push the listeners down from the java client to the webclient, or will you have to send every event from the webclient to the java client to before characters can be rendered?
> Thanks,
> <customer>
>
> On 6 August 2014 15:20, Greg Shah <ges@goldencode.com> wrote:
>
> <customer>,
>
> > In our hangout, I believe you mentioned that the java client doesn't do too much in the way of Progress processing - it basically just renders the screens, and passes UI events back to the P2J server.
>
> Yes, although this is a relative thing. Compared to the 4GL logic/database processing on the server side, there is very little "processing" being done on the client.
>
> However, there is a tremendous amount of Progress complexity in the UI itself. For example, we handle all data formatting for the UI at the client-side. This means that when a FILL-IN is displaying a date, that widget "knows" its format string and the date value being displayed/edited and handles all that 4GL behavior properly, without the server being involved. Although there is no converted application code processing there, the "thin client" is not as thin as we would wish.
>
> > However, I believe you mentioned something about the character client holding the screen buffer and passes this back to P2J on commit.
>
> The Progress single-threaded design comes into play here. Since all 4GL code (including UI) can only ever run on a single thread, we are able to reasonably split the processing of the Progress fat client into 2 parts:
>
> - business logic, database and UI controller on the server
> - native platform access and UI presentation/interaction engine on the client
>
> This works because when we reach some point in the server-side processing such that a client-side feature must be invoked, we can synchronously call over (via our RemoteObject "RPC") to the client and let that code work, while the server blocks waiting for the response.

On the client, if we hit an event that should fire something on the server-side (e.g. a trigger or a validation expression) then instead of "returning" from the server's request, we just call over to the server to get that functionality. With nested WAIT-FORs (using triggers), this means that we can nest calls from server to client to server to client up to an arbitrary depth. Eventually, these will start returning and ultimately when the user is done with whatever original interaction was being executed in the UI, the original call from the server will return. At that point the business logic continues on the server-side where it was originally blocked.

Thus at any point in time only the server or the client can be executing, while the other is blocked. We call this a "conversation mode" in our protocol.

The way we do this without any converted code being used on the client is that we send data to the client to configure it:

- frame definitions (the layout/structure/configuration of each frame and its contained widgets)
- screen buffers (the state of each widget in the frame, including the data, any edits, whether the widget has been ENTERED...)

When the server invokes some method on the client (e.g. DISPLAY is a ThinClient.display() method call), there will be some data passed as parameters. Each method has the data necessary. Any 4GL feature like DISPLAY that operates on the screen buffer, will take the current copy of the screen buffer from the server. Likewise, whenever the client returns from a call that can change the screen buffer or calls over to the server to invoke functionality that could read the latest screen buffer (e.g. both triggers and validate expressions can do this), the latest screen buffer will be returned/sent.

Since processing only ever occurs on one side (server or client) at a time, we always know which side has the authoritative version of the screen buffer and we can easily synchronize it by sending it along with the methods calls/returns.

> I've just realised, however, that our (GUI) Progress code contains many references to INPUT-VALUE or SCREEN-VALUE of widgets, which are the current contents of the screen buffer. (This is similar to using the INPUT <field> keyword in ChUI). So therefore, when executing that code, the P2J server will need to know the current screen contents of those fields.

Yes, we fully support these features (which are only ever implemented on the server) and as noted above, the screen buffer will always reflect the latest state, so they will work as expected.

> And if it's not too complicated, can you confirm how the SCREEN-VALUE impacts on the architecture webclient?

I don't think there is much of an impact here. The web driver will be operating at a much lower level in the client. While it will be receiving data and sending edits back, it won't be dealing directly with a screen buffer. That screen buffer logic will remain at the higher level of the client and will be common code.

I hope this helps. Let me know if I need to clarify something or if you have other questons.

Greg
On 08/06/2014 09:03 AM, <customer> wrote:

> Hi Greg,
> Would you mind clarifying something for me please...
> In our hangout, I believe you mentioned that the java client doesn't do too much in the way of Progress processing - it basically just renders the screens, and passes UI events back to the P2J server. However, I believe you mentioned something about the character client holding the screen buffer and passes this back to P2J on commit.
> I've just realised, however, that our (GUI) Progress code contains many references to INPUT-VALUE or SCREEN-VALUE of widgets, which are the current contents of the screen buffer. (This is similar to using the INPUT <field> keyword in ChUI). So therefore, when executing that code, the P2J server will need to know the current screen contents of those fields.
> So, does that mean I misunderstood what you meant about the java client holding the screen buffer? Or do you hold the screen buffer in both places (on the P2J server, and the java client), and somehow sync between the two? E.g. when SCREEN-VALUE/INPUT-VALUE is referenced?
> And if it's not too complicated, can you confirm how the SCREEN-VALUE impacts on the architecture webclient?
> Thanks,
> <customer>

#2 - 03/23/2016 05:21 PM - Greg Shah

- Target version changed from Milestone 12 to Milestone 16

#3 - 07/29/2016 09:42 AM - Greg Shah

- Assignee set to Eugenie Lyzenko

- % Done changed from 0 to 100

- Status changed from New to Closed

The issues described in this task are completely resolved by the changes in task branch 3051d revision 11086. Those changes will be tested and merged to trunk in #3051. As such, I'm closing this task.

#4 - 08/05/2016 07:39 AM - Greg Shah

3051d was merged to trunk as rev 11077.

#5 - 11/16/2016 12:22 PM - Greg Shah

- Target version changed from Milestone 16 to Cleanup and Stabilization for GUI