User Interface - Bug #2914

Bug # 2677 (New): fix drawing and functional differences between P2J GUI and 4GL GUI

Fix errors in window_sizing/test_runner.p

12/03/2015 06:03 PM - Hynek Cihlar

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Hynek Cihlar	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	GUI Support for a Complex ADM2 App		
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

History

#1 - 02/16/2016 09:59 AM - Greg Shah

Please create a 2914a task branch for the changes. I'd also like to see some details posted here about the problems you are fixing.

#2 - 02/16/2016 11:19 AM - Hynek Cihlar

Greg Shah wrote:

> Please create a 2914a task branch for the changes. I'd also like to see some details posted here about the problems you are fixing.

So far I have no p2j changes. I've been resolving unexpected errors (cca 20 occurences) when the window_sizing test suite is run in the native 4GL environment. Some of the errors are caused by different Windows scheme settings. I am incorporating these scheme variables into the test cases.

#3 - 02/17/2016 02:06 PM - Hynek Cihlar

I've created task branch 2914a.

#4 - 02/19/2016 07:37 AM - Hynek Cihlar

Rebased task branch 2914a against trunk revision 10972. 2914a is now at revision 10974.

#5 - 02/19/2016 09:12 AM - Hynek Cihlar

I have found three conversion errors related to type conversion.

Case 1:

```
DEFAULT-WINDOW:FULL-WIDTH-CHARS = 1 NO-ERROR.
```

Converts to:

```
defaultWindow().unwrap().readOnlyError(new decimal("FULL-WIDTH-CHARS"));
```

Case 2:

```
function foo returns decimal (input i as decimal):
   return 1.
end.

foo(20).
```

Converts to:

```
   public decimal foo(final decimal _i)
   {
      return function("foo", decimal.class, new Block()
      {
         decimal i = new decimal(_i);

         public void body()
         {
            returnNormal(1);
         }
      });
   }

foo(new integer(20));
```

Case 3:

```
function foo returns decimal (input i as decimal):
   return 1.
end.

foo(SESSION:WIDTH-PIXELS).
```

Converts to:

```
   public decimal foo(final decimal _i)
   {
      return function("foo", decimal.class, new Block()
      {
         decimal i = new decimal(_i);

         public void body()
         {
            returnNormal(1);
         }
      });
   }

foo(LogicalTerminal.getWidthPixels());
```

Since the errors blocked me, I went ahead and fixed cases 1 and 2. Number 3 seems to be similar to number 2, so I plan to fix it accordingly.

Please review task branch 2914a revision 10974 with fixes of cases 1 and 2.

#6 - 02/19/2016 01:36 PM - Greg Shah

Code Review Task Branch 2914a Revision 10974

The literals.rules change is fine.

The functions_procedures.rules change doesn't look correct. At that point in the code, copy is known to be type == prog.assign and pref will always be the 2nd child of that prog.assign node.

The original code resets pref to be the first child of pref which is the grandchild of the prog.assign node, through its right operand (its 2nd child). I see that we are trying to force that node to be a prog.expression which means that it must not already be a prog.expression in all cases (but it often will be). Depending on the type of that right operand when it is a non-expression type, I can see that the code might fail. For example, perhaps it doesn't have a child node at all.

But I can't really understand your change. It will always reset pref to be the same thing as pref. In other words, in the normal case where we do expect there to be a grandchild node, that node won't be set to be the prog.string. Instead, the new code is forcing the node that was just set as a prog.expression to now be a prog.string. There will never be an expression node with these changes. That seems wrong.

#7 - 02/19/2016 04:37 PM - Constantin Asofiei

Hynek, the first case looks like an incorrect wrapping; the AST for the right-side value has a no_wrap=true annotation, but is not considered, as the c'tor is already emitted. The c'tor is emitted by this code in convert/assignments.rules:355:

```
      <!-- wrap results that will be assigned to writable attribute  -->
      <rule>type == prog.expression       and
            parent.type == prog.assign    and
            parent.firstChild.type == prog.colon

         <action>exprType = ecw.expressionType(copy.getChildAt(0), true)</action>
         <action>classname = ecw.expressionType(parent.getChildAt(0), true)</action>

         <rule>classname != null and !evalLib("is_type_assignable", classname, exprType)
            <action>createPeerAst(java.constructor, classname, closestPeerId)</action>
         </rule>
      </rule>
```

As you can see, if types differ, we force a wrapping regardless of no_wrap annotation.

#8 - 02/19/2016 04:48 PM - Constantin Asofiei

From previous note, changing the code to:

```
      <!-- wrap results that will be assigned to writable attribute  -->
      <rule>type == prog.expression       and
            parent.type == prog.assign    and
            parent.firstChild.type == prog.colon
         <action>ref = parent.firstChild</action>
         <action>ref = ref.getChildAt(1)</action>

         <rule>!evalLib("read_only_attribute", ref)
            <action>exprType = ecw.expressionType(copy.getChildAt(0), true)</action>
            <action>classname = ecw.expressionType(parent.getChildAt(0), true)</action>

            <rule>classname != null and !evalLib("is_type_assignable", classname, exprType)
               <action>createPeerAst(java.constructor, classname, closestPeerId)</action>
            </rule>
         </rule>
      </rule>
```

will avoid wrapping read-only attributes.

#9 - 02/19/2016 04:57 PM - Constantin Asofiei

For the third case, the chp_wrapper=decimal annotation is emitted at the COLON node:

```
    <ast col="0" id="219043332139" line="0" text="assignment" type="ASSIGNMENT">
      <ast col="0" id="219043332140" line="0" text="expression" type="EXPRESSION">
        <annotation datatype="java.lang.Long" key="peerid" value="236223201344"/>
        <ast col="1" id="219043332141" line="9" text="foo" type="FUNC_DEC">
          <annotation datatype="java.lang.Long" key="oldtype" value="2405"/>
          <annotation datatype="java.lang.Long" key="refid" value="219043332112"/>
          <annotation datatype="java.lang.Boolean" key="ignorecast" value="true"/>
          <annotation datatype="java.lang.String" key="casttype" value="decimal"/>
          <annotation datatype="java.lang.Long" key="peerid" value="236223201345"/>
          <annotation datatype="java.lang.Boolean" key="wrap" value="true"/>
          <ast col="12" id="219043332143" line="9" text=":" type="COLON">
            <annotation datatype="java.lang.String" key="chp_wrapper" value="decimal"/>
            <annotation datatype="java.lang.Long" key="peerid" value="236223201346"/>
            <ast col="5" id="219043332144" line="9" text="session" type="SYS_HANDLE">
              <annotation datatype="java.lang.Long" key="oldtype" value="805"/>
              <annotation datatype="java.lang.Long" key="tempidx" value="256"/>
            </ast>
            <ast col="13" id="219043332145" line="9" text="width-pixels" type="ATTR_INT">
              <annotation datatype="java.lang.Long" key="oldtype" value="2321"/>
            </ast>
          </ast>
        </ast>
      </ast>
    </ast>
```

I think this needs to be solved at the methods_attributes.rules, as this is not aware of the annotation. Adding this code at line 3809 solves the case:

```
<rule>isNote("chp_wrapper")
   <action>
      createPeerAst(java.constructor, getNoteString("chp_wrapper"), closestPeerId)
   </action>
</rule>
```

These two and your literals.rules change from the branch solve all the 3 cases you pointed. Although I haven't tested more complex cases, or for regressions.

#10 - 02/19/2016 07:13 PM - Hynek Cihlar

Thanks for the inputs. I have reverted my changes except of the fix of Case 1 and applied Constantin's changes. The result is in task branch 2914a revision 10975. I have started regression testing on the revision.

#11 - 02/20/2016 02:26 PM - Hynek Cihlar

There is a technical limit of the max window size in Swing GUI driver. The screen buffer implemented by BufferedImage and allocated in SwingEmulatedWindow.resizeScreenBuffers() has a cap at Integer.MAX_VALUE of pixel area. This translates to a square with root of 46340.

I have implemented a workaround fix in 2914a revision 10979. The fix adjusts the window size during window layout to fit into the driver's limit.

Also, with the default client Java heap setup, there are out of memory exceptions when attempting to resize the window to the upper limits.

#12 - 02/22/2016 08:40 AM - Greg Shah

> Also, with the default client Java heap setup, there are out of memory exceptions when attempting to resize the window to the upper limits.

What heap is needed to avoid the OOMEs?

#13 - 02/22/2016 11:47 AM - Hynek Cihlar

Greg Shah wrote:

> Also, with the default client Java heap setup, there are out of memory exceptions when attempting to resize the window to the upper limits.

What heap is needed to avoid the OOMEs?

For the max window size (as limited by Swing GUI driver, width x height == Integer.MAX_VALUE) it is around 22 GB of heap space to avoid the OOMEs, and around 9 GB of heap space for half the max window size.

#14 - 02/23/2016 10:19 AM - Greg Shah

Yikes!

I think we need to put some basic protections into place as a default. The customer should be able to override those protections in order to do something unusual (like running the window_sizing/ testcases). But by default, we should not get into a situation where 4GL code will cause OOMEs.

In practical terms, the Swing window itself never really needs to be so large that it uses GBs of memory. Perhaps we can set a limit that should display even very large windows but still be a reasonable heap size in Java. If possible we would allow assignment and reporting of the huge sizes in the converted 4GL code, but the actual window size would be limited (unless overridden by the customer).

#15 - 02/23/2016 12:34 PM - Hynek Cihlar

Greg Shah wrote:

> Yikes!
>
> I think we need to put some basic protections into place as a default. The customer should be able to override those protections in order to do something unusual (like running the window_sizing/ testcases). But by default, we should not get into a situation where 4GL code will cause OOMEs.
>
> In practical terms, the Swing window itself never really needs to be so large that it uses GBs of memory. Perhaps we can set a limit that should display even very large windows but still be a reasonable heap size in Java. If possible we would allow assignment and reporting of the huge sizes in the converted 4GL code, but the actual window size would be limited (unless overridden by the customer).

I have implemented the enforcement of the max physical Swing window size. I can make the limit configurable and make the app logic report the assigned (and possibly larger size) as you suggested.

Another solution I see would be to avoid double-buffering from a certain (configurable) window size. We would simply avoid BufferedImage when the window size grew larger then a specified limit. Yes this would introduce flickering, but we would get at least some functionality and avoid the potential OOMEs.

The last solution would be to implement our own BufferedImage employing a lossless compression. But this would probably require significantly more effort.

#16 - 02/24/2016 07:37 AM - Greg Shah

> I can make the limit configurable and make the app logic report the assigned (and possibly larger size) as you suggested.

Yes, sounds good.

> Another solution I see would be to avoid double-buffering from a certain (configurable) window size. We would simply avoid BufferedImage when the window size grew larger then a specified limit. Yes this would introduce flickering, but we would get at least some functionality and avoid the potential OOMEs.

I like this, so long as it doesn't make things so complicated that they become fragile.

#17 - 02/24/2016 07:40 AM - Constantin Asofiei

Hynek Cihlar wrote:

> Another solution I see would be to avoid double-buffering from a certain (configurable) window size. We would simply avoid BufferedImage when the window size grew larger then a specified limit. Yes this would introduce flickering, but we would get at least some functionality and avoid the potential OOMEs.
>
> The last solution would be to implement our own BufferedImage employing a lossless compression. But this would probably require significantly more effort.

Here you are referring to the Swing client, right? For the web client, I think we need to find the maximum window size which can still be manageable by the browser, without freezing it... considering that for each window there are two canvas instances.

#18 - 02/24/2016 08:14 AM - Hynek Cihlar

Greg Shah wrote:

> > I can make the limit configurable and make the app logic report the assigned (and possibly larger size) as you suggested.

> Yes, sounds good.
>
> > Another solution I see would be to avoid double-buffering from a certain (configurable) window size. We would simply avoid BufferedImage when the window size grew larger then a specified limit. Yes this would introduce flickering, but we would get at least some functionality and avoid the potential OOMEs.
>
> I like this, so long as it doesn't make things so complicated that they become fragile.

I think it should be relatively simple. The most problematic part will be triggering repaint when the unbuffered window is moved to the area not drawn, in this case widget repaint will have to be triggered from the driver.

I'll go with this approach, if I hit any complications I will report back.

#19 - 02/24/2016 08:44 AM - Hynek Cihlar

Constantin Asofiei wrote:

> Hynek Cihlar wrote:
>
> > Another solution I see would be to avoid double-buffering from a certain (configurable) window size. We would simply avoid BufferedImage when the window size grew larger then a specified limit. Yes this would introduce flickering, but we would get at least some functionality and avoid the potential OOMEs.
> >
> > The last solution would be to implement our own BufferedImage employing a lossless compression. But this would probably require significantly more effort.
>
> Here you are referring to the Swing client, right?

Yes.

> For the web client, I think we need to find the maximum window size which can still be manageable by the browser, without freezing it... considering that for each window there are two canvas instances.

Right, according to this link http://stackoverflow.com/questions/6081483/maximum-size-of-a-canvas-element?rq=1 there are size canvas limits we can reach.

#20 - 02/26/2016 09:33 AM - Hynek Cihlar

I have committed the solution for the large window sizes in Swing, see task branch 2914a revision 10984.

I improved a bit the original idea with limiting the screen buffer size. The screen buffer is allocated only for the window's visible area plus some overlap in case the window is bigger than the screen size. Because the buffer is allocated only for a portion of the window area its origin must be adjusted when the window is moved. The overlap allows the window to be user-moved smoothly, the buffer must be readjusted and window redrawn only when the buffer boundary is hit.

#21 - 02/26/2016 09:58 AM - Greg Shah

I will do a code review shortly.

What is left to do on this task?

Have you handled the web client as well?

#22 - 02/26/2016 10:28 AM - Hynek Cihlar

Greg Shah wrote:

> I will do a code review shortly.
>
> What is left to do on this task?

There are some unresolved unexpected results in the window_sizing suite to be cleared. I will start working on these when I finish testing the web client large window size.

> Have you handled the web client as well?

This is currently in progress. I've had some troubles starting the web client.

#23 - 02/26/2016 10:38 AM - Greg Shah

Code Review Task Branch 2914a Revision 10984

1. Is the ClientBuilder.localStart() change (to redirect STDOUT) something that can stay permanently? Or does it affect the spawned client environment in a negative way?

2. I realize that you may still be planning to add history entries in some places. But I think it is useful to know that SwingEmulatedWindow.getVirtualScreenSize() needs javadoc.

3. I think it is worth an early run of regression testing to ensure that there isn't something important that will require an extended cleanup period.

#24 - 02/26/2016 10:56 AM - Hynek Cihlar

Greg Shah wrote:

> Code Review Task Branch 2914a Revision 10984
>
> 1. Is the ClientBuilder.localStart() change (to redirect STDOUT) something that can stay permanently? Or does it affect the spawned client environment in a negative way?

This is a debug change that will go away.

> 2. I realize that you may still be planning to add history entries in some places. But I think it is useful to know that SwingEmulatedWindow.getVirtualScreenSize() needs javadoc.

Yes I haven't been adding all file history entries, but I missed the javadoc for SwingEmulatedWindow.getVirtualScreenSize(). I will add it.

> 3. I think it is worth an early run of regression testing to ensure that there isn't something important that will require an extended cleanup period.

Yes, it is running.

#25 - 02/26/2016 10:59 AM - Hynek Cihlar

Greg Shah wrote:

> What is left to do on this task?

I also found two new trunk regressions.

- Window cannot be user-moved out of the screen rectangle.
- When moving the window by mouse the cursor doesn't keep the window mouse-down location.

#26 - 02/26/2016 07:51 PM - Hynek Cihlar

Task branch 2914a revision 10984 passed conversion testing.

#27 - 02/26/2016 09:10 PM - Hynek Cihlar

I can confirm the window size performance problem exists in Web GUI, too. I was able to reach the max window size of about 5000 by 5000 pixels, higher values froze the browser and the system and eventually triggered the "Unresponsive script" dialog and caused unexpected errors.

#28 - 02/27/2016 11:09 AM - Hynek Cihlar

Hynek Cihlar wrote:

> Greg Shah wrote:
>
> > What is left to do on this task?
>
> I also found two new trunk regressions.
>
> - Window cannot be user-moved out of the screen rectangle.

Interestingly, this is not a regression but a feature of Swing/AWT.

When the window fits fully on the screen, calling java.awt.Window.setLocation() doesn't allow the window to move outside of the screen bounds. So for example when window.setLocation(-10, -10) is called, it will end up at actual location 0,0.

But when the window is resized so that it doesn't fit on the screen, java.awt.Window.setLocation() will allow to set off-screen locations. An example, window height is set to be greater than screen height, width is set less than screen height, a call to window.setLocation(-10, -10) will position the window at -10,-10.

At this moment I don't see an easy workaround. As long as we draw the window decorations ourselves we have to use window.setLocation() to reposition the window on mouse drag.

Greg, what do you think?

#29 - 02/27/2016 11:14 AM - Hynek Cihlar

Hynek Cihlar wrote:

> Hynek Cihlar wrote:
>
> > Greg Shah wrote:
> >
> > > What is left to do on this task?
> >
> > I also found two new trunk regressions.
> >
> > - Window cannot be user-moved out of the screen rectangle.
>
> Interestingly, this is not a regression but a feature of Swing/AWT.
>
> When the window fits fully on the screen, calling java.awt.Window.setLocation() doesn't allow the window to move outside of the screen bounds. So for example when window.setLocation(-10, -10) is called, it will end up at actual location 0,0.
>
> But when the window is resized so that it doesn't fit on the screen, java.awt.Window.setLocation() will allow to set off-screen locations. An example, window height is set to be greater than screen height, width is set less than screen height, a call to window.setLocation(-10, -10) will position the window at -10,-10.
>
> At this moment I don't see an easy workaround. As long as we draw the window decorations ourselves we have to use java.awt.Window.setLocation() to reposition the window on mouse drag.
>
> Greg, what do you think?

I should also mention that this behavior is OS-dependent. For example on Windows java.awt.Window.setLocation() will set the off-screen location ok.

#30 - 02/29/2016 10:11 AM - Hynek Cihlar

Rebased task branch 2914a against trunk revision 10974. 2914a is now at revision 10988.

#31 - 02/29/2016 12:18 PM - Greg Shah

> When the window fits fully on the screen, calling java.awt.Window.setLocation() doesn't allow the window to move outside of the screen bounds. So for example when window.setLocation(-10, -10) is called, it will end up at actual location 0,0.
>
> But when the window is resized so that it doesn't fit on the screen, java.awt.Window.setLocation() will allow to set off-screen locations. An example, window height is set to be greater than screen height, width is set less than screen height, a call to window.setLocation(-10, -10) will position the window at -10,-10.
>
> At this moment I don't see an easy workaround. As long as we draw the window decorations ourselves we have to use java.awt.Window.setLocation() to reposition the window on mouse drag.

Other than the window being displayed at the wrong location (and thus being more visible than expected), is there any other deviation in the behavior?

Please document this difference in #2961. Also, please create a new task for this that we can look at later, if we choose to try to address this.

#32 - 02/29/2016 02:17 PM - Hynek Cihlar

Greg Shah wrote:

> > When the window fits fully on the screen, calling java.awt.Window.setLocation() doesn't allow the window to move outside of the screen bounds. So for example when window.setLocation(-10, -10) is called, it will end up at actual location 0,0.
> >
> > But when the window is resized so that it doesn't fit on the screen, java.awt.Window.setLocation() will allow to set off-screen locations. An example, window height is set to be greater than screen height, width is set less than screen height, a call to window.setLocation(-10, -10) will position the window at -10,-10.
> >
> > At this moment I don't see an easy workaround. As long as we draw the window decorations ourselves we have to use java.awt.Window.setLocation() to reposition the window on mouse drag.
>
> Other than the window being displayed at the wrong location (and thus being more visible than expected), is there any other deviation in the behavior?

I didn't find any other deviation related to this issue.

> Please document this difference in #2961. Also, please create a new task for this that we can look at later, if we choose to try to address this.

Roger.

#33 - 03/01/2016 09:27 AM - Hynek Cihlar

In 4GL window HEIGHT attribute can be assigned to smaller values than the smallest physical window size. So we end up with a mismatch between what is reported and what is actually displayed on screen. Fixing this properly will require a bit more time because in general we always match the reported widget values with the actual state.

As this IMHO has small negative impact I would create new issue and deal with this later. Greg, what do you think?

#34 - 03/01/2016 09:49 AM - Greg Shah

> As this IMHO has small negative impact I would create new issue and deal with this later. Greg, what do you think?

Yes.

#35 - 03/03/2016 03:46 PM - Hynek Cihlar

During frame setup when a client remote call is performed the frame may be considered dead and removed from the client.

I stepped on this when I added LT.getClient().isRedirected() to BaseEntity.setColumnOrRowWorker(). The call causes execution of LT.getChanges() which goes through all frames and marks all with no valid instantiating procedure as dead. Because the subjected frame was in setup phase, it was marked as dead.

Please review the fix in 2914a revision 10991.

#36 - 03/03/2016 04:48 PM - Hynek Cihlar

Rebased task branch 2914a against trunk revision 10978. 2914a is now at revision 10996.

#37 - 03/04/2016 08:57 AM - Hynek Cihlar

Please review 2914a rev 10997. It is a release candidate with all the required changes for this issue.
Note that ChUI regression test run is in progress waiting for a pass as well as GUI regression testing is in progress.

#38 - 03/04/2016 04:16 PM - Greg Shah

Code Review Task Branch 2914a Revision 10997

This revision is not compilable.

1. There are merge problems with the history entries in methods_attributes.rules, LogicalTerminal.

2. There are merge problems with the code in BaseEntity.setXOrYWorker(), WindowTitlebar.minimumSize(), SwingEmulatedWindow.repositionScreenBuffers(), SwingEmulatedWindow.getVirtualScreenSize().

3. MouseMovable.updateWidgetLocation() is missing javadoc.

#39 - 03/04/2016 04:43 PM - Hynek Cihlar

Greg Shah wrote:

> Code Review Task Branch 2914a Revision 10997
>
> This revision is not compilable.
>
> 1. There are merge problems with the history entries in methods_attributes.rules, LogicalTerminal.
>
> 2. There are merge problems with the code in BaseEntity.setXOrYWorker(), WindowTitlebar.minimumSize(), SwingEmulatedWindow.repositionScreenBuffers(), SwingEmulatedWindow.getVirtualScreenSize().

Greg, I can compile the revision ok, even on devsrv01. Try to bzr revert.

#40 - 03/04/2016 04:49 PM - Greg Shah

Code Review Task Branch 2914a Revision 10997

You're right, I had not reverted. It was my mistake, sorry.

Please note that MouseMovable.updateWidgetLocation() is missing javadoc. Otherwise the changes look fine.

#41 - 03/04/2016 04:53 PM - Hynek Cihlar

Greg Shah wrote:

> Code Review Task Branch 2914a Revision 10997
>
> You're right, I had not reverted. It was my mistake, sorry.

No problem at all.

I have committed 2914a rev 10998. Fixed a GUI regression - wrong minimum size for modal windows and overlay windows. Also added missing javadoc to MouseMovable.updateWidgetLocation().

The revision has passed GUI regression tests.

#42 - 03/04/2016 04:56 PM - Hynek Cihlar

2914a passed ChUI regression tests. Note that the regression fix in 10998 only affects GUI code.

#43 - 03/04/2016 04:58 PM - Greg Shah

Code Review Task Branch 2914a Revision 10998

The changes are good.

Please merge this to trunk.

#44 - 03/04/2016 05:10 PM - Hynek Cihlar

2914a committed to trunk as rev 10979 and archived.

#45 - 03/04/2016 05:12 PM - Greg Shah

I can close this task, right?

#46 - 03/04/2016 05:26 PM - Hynek Cihlar

Greg Shah wrote:

> I can close this task, right?

Yes, please.

#47 - 03/04/2016 05:27 PM - Greg Shah

- Assignee set to Hynek Cihlar

- Status changed from New to Closed

- Target version set to Milestone 12

- Start date deleted (12/03/2015)

- % Done changed from 0 to 100

#48 - 03/07/2016 05:56 PM - Hynek Cihlar

I have opened task branch 2914b for regression fixes.

#49 - 03/08/2016 11:27 AM - Hynek Cihlar

Please review task branch 2914b revision 10981. It contains a regression fix - frame/window size may be calculated wrong.

ChUI regression tests in progress.

#50 - 03/08/2016 12:06 PM - Greg Shah

Code Review Task Branch 2914b Revision 10981

The changes are fine.

Please note that a rebase will be needed.

#51 - 03/08/2016 12:08 PM - Hynek Cihlar

Greg Shah wrote:

> Please note that a rebase will be needed.

Right, thanks for pointing this out.

#52 - 03/08/2016 12:14 PM - Hynek Cihlar

Rebased task branch 2914b against trunk revision 10980. 2914b is now at revision 10982. ChUI regression test restarted.

#53 - 03/10/2016 07:28 PM - Hynek Cihlar

Rebased task branch 2914b against trunk revision 10981. 2914b is now at revision 10985.

#54 - 03/13/2016 10:34 AM - Hynek Cihlar

Rebased task branch 2914b against trunk revision 10982. 2914b is now at revision 10986.

#55 - 03/13/2016 11:26 AM - Hynek Cihlar

Please review 2914b revision 10987. The branch resolves a GUI regression of wrong frame size and also fixes frame scroll layout calculation.

ChUI regression test in progress. Note that not all GUI regression tests were performed due to an existing GUI regression.

#56 - 03/15/2016 08:08 AM - Constantin Asofiei

Hynek Cihlar wrote:

> Please review 2914b revision 10987.

I'm OK with the changes.

Please use this patch (from 2794b) to check the other GUI tests:

```
Index: src/com/goldencode/p2j/ui/GenericFrame.java
===================================================================
--- src/com/goldencode/p2j/ui/GenericFrame.java    (revision 753)
+++ src/com/goldencode/p2j/ui/GenericFrame.java    (working copy)
@@ -779,6 +779,8 @@
 ** 321 EVL 20160224          Javadoc fixes to make compatible with Oracle Java 8 for Solaris 10.
 ** 322 CA  20160303          Made attribute setters be a no-op if the value doesn't change.
 ** 323 OM  20160205          Matching frames conditions were incorrect. Added cleanup on error.
+** 324 CA  20160315          Fixed H323 regression in initLiteralFG - an empty array needs to be
+**                           passed to wrapWidgetsToFieldGroup, and not the current widget list.
 */

 package com.goldencode.p2j.ui;
@@ -9881,7 +9883,7 @@
     */
    private FieldGroup initLiteralFG()
    {
-      return wrapWidgetsToFieldGroup(this.widgets, null);
+      return wrapWidgetsToFieldGroup(new GenericWidget[0], null);
    }
```

```
    /**
```

#57 - 03/15/2016 01:06 PM - Hynek Cihlar

Constantin Asofiei wrote:

> Hynek Cihlar wrote:
>
> > Please review 2914b revision 10987.
>
> I'm OK with the changes.
>
> Please use this patch (from 2794b) to check the other GUI tests:
> [...]

2914b with the patch passes GUI regression tests. Also, ChUI regression tests have passed.

Constantin, are you gonna merge the patch to trunk?

#58 - 03/15/2016 01:12 PM - Constantin Asofiei

Hynek Cihlar wrote:

> Constantin, are you gonna merge the patch to trunk?

Yes, after it finishes the main runtime testing part.

#59 - 03/16/2016 05:03 AM - Hynek Cihlar

Rebased task branch 2914b against trunk revision 10983. 2914b is now at revision 10989. 2914b is ready to be merged to trunk.

#60 - 03/17/2016 04:48 PM - Greg Shah

Please merge 2914b to trunk.

#61 - 03/18/2016 04:18 AM - Hynek Cihlar

Task branch 2914b was merged to trunk (rev 10984) and archived.

#62 - 11/16/2016 12:12 PM - Greg Shah

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