User Interface - Feature #2476

Feature # 2252 (Closed): implement GUI client support

window icon support improvements

01/06/2015 05:36 PM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Eugenie Lyzenko	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:	GUI Support for a Complex ADM2 App		
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to User Interface - Bug #2792: image loading issues	**Closed**

History

#1 - 01/06/2015 05:39 PM - Greg Shah

1. Loading icons using using a handle that isn't CURRENT-WINDOW does not work.

2. Add Window.loadSmallIcon() runtime support.

3. By default, the Window should display the Golden Code icon. Right now, it only shows that icon as a fall-back when the converted code tries to load other icons and fails.

#2 - 01/06/2015 05:40 PM - Greg Shah

- Parent task set to #2252

#3 - 01/07/2015 08:02 PM - Eugenie Lyzenko

- File image5-4gl.jpg added

- File image5-p2j.jpg added

- File evl_upd20150107b.zip added

1. Loading icons using using a handle that isn't CURRENT-WINDOW does not work.

The main issue with loading icons for dynamically created windows is the new window is not showing. This happens because the window simulator was not registered with screen driver for dynamic window. We need to add respective code into WindowGuiImpl.initialize() method. With such modification the behavior looks correct for icon loading. Compare screen shots from 4GL and P2J for the following code sample:

```
...
message "Hit a key to start".
pause.

def var hwin as handle.
create window hwin
   assign width-chars  = 40
          height-chars = 20
          row          = 10
```

```
          column       = 50.

hwin:load-icon("customer_4bpp.ico").
hwin:visible = true.

message "Application completed. Press a key.".
pause.
...
```

The attached update for review fixes this issue.

The question remaining unclear is the approach we need to take to position the new window on the screen. The window can not be related to some P2J container. It locates on system desktop. Will we let the OS to position the P2J window in this case? Or something else? Need to investigate how 4GL on Window implements this. Currently P2J places new window behind the main one so if main window is not moved manually - we do not even see the new window and this can become a problem for user.

#4 - 01/07/2015 08:22 PM - Eugenie Lyzenko

Also note the new window in 4GL is placed over the main one, while in P2J the main window remains on top. This is the difference.

#5 - 01/08/2015 04:58 AM - Constantin Asofiei

Eugenie, your changes seem OK, but I'm not sure when the GUI code broke, as something simple as this doesn't work anymore without your update:

```
create window h.
h:title = "bla".
h:visible = true.
```

#6 - 01/08/2015 05:25 AM - Greg Shah

I don't know that we need to runtime test this since from a ChUI perspective this just adds a no-op method. Thoughts?

#7 - 01/08/2015 05:26 AM - Constantin Asofiei

Greg Shah wrote:

> I don't know that we need to runtime test this since from a ChUI perspective this just adds a no-op method. Thoughts?

Correct, no runtime testing is needed.

#8 - 01/08/2015 05:28 AM - Greg Shah

Eugenie: OK, go ahead and check it in.

#9 - 01/08/2015 08:02 AM - Eugenie Lyzenko

The update evl_upd20150107b.zip has been committed in bzr as 10699.

#10 - 01/08/2015 10:05 PM - Eugenie Lyzenko

2. Add Window.loadSmallIcon() runtime support.

About LOAD-SMALL-ICON. Generally speaking there is no difference between small and regular icon size from the title bar drawing perspective. The size is the same as the title bar height. The only thing we need to consider is the small icon loading prefers the 16x16 icon version if the icon file has multiple icons with different sizes and 32x32 if this is the only icon the file has. Like this regular loading prefers 32x32 size icon version if icon file has multiple versions and use 16x16 if regular size icon not found.

So we need to implement the way we extract respective image data from file that has several images. Currently we load icon file as a byte array and use the first available icon image data.

Moreover according to 4GL document icon size difference should work in Windows 95 or NT method). So I do not even sure we need to implement this feature et all.

#11 - 01/09/2015 07:54 AM - Greg Shah

Please make sure you support all the variants of these methods (with more than one parameter).

In addition, it is important to test the following cases:

- different propaths
- relative paths prefixed on the icon name which specify a file that can only be matched by a propath search + the relative path
- relative paths prefixed on the icon name which specify a file that can only be matched by the current directory + the relative path
- relative paths prefixed on the icon name which specify a file that can be matched by either the current directory or soemthing in the propath + the relative path
- absolute paths

#12 - 01/09/2015 07:55 AM - Greg Shah

> Moreover according to 4GL document icon size difference should work in Windows 95 or NT method). So I do not even sure we need to implement this feature et all.

If you can detect a difference when running different cases in the 4GL then we must also implement the same difference.

#13 - 01/09/2015 01:24 PM - Eugenie Lyzenko

- File evl_upd20150109a.zip added

This upload for review implements small/regular icon loading. The idea is the difference we need to make is the image to choose from icon file. The approach offering is if the icon file contains multiple icons - we search first available icon with respective size. If required size can not be found - we will use the first available icon from file. This is how 4GL currently works. BTW the Windows 7 also support LOAD-SMALL-ICON method.

Also contains fix for missed default gclogo.ico file loading from JAR for window.

While you review the update I'm going to test different path combinations for loaded icon cases.

#14 - 01/09/2015 07:36 PM - Eugenie Lyzenko

- File evl_upd20150109b.zip added

This update for review adds current path adding to be included in image loading approach. If the image or icon was not found in PROPATH the LT attempts to load file from the current directory or sub-paths related to the current directory. So with this update the icon loading works for the cases listed above and if there are no notes we can start regression testing.

#15 - 01/10/2015 01:07 PM - Greg Shah

Code Review evl_upd20150109b.zip

1. In ImageGuiImpl, should line 258 be if (readers.hasNext()) or should it be while (readers.hasNext())?

2. It seems that both loadIcon() and loadSmallIcon()@ should actually report if there is some failure (e.g. the icon could not be found). Doesn't Progress return false if the icon is not found? See below.

3. We need to step back and clean up our general approach to image/icon loading. I think our current implementation has been done in a rush and could use from improvement. The problems were there before your recent changes, so this is not strictly feedback on your changes, but rather these are general improvements that we need to make:

- We have the following possible sources of images/icons (in this order of loading precedence):
  1. application jar file (NOT p2j.jar), this is a server-side source
  2. client-side file system (found via propath search if it is a relative name)
  3. the gclogo.ico from the client-side p2j.jar
- The ICON attribute of a window is read-only, so the only way to associate an icon with a window is via LOAD-ICON() or LOAD-SMALL-ICON(). Before they are called, the **server-side window** (not the client-side WindowTitleBar) must look to the following sources to set the default icon:
  1. Any value specified in the "window-icon" directory entry.
  2. The gclogo.ico, which can be loaded from the client-side p2j.jar.
- Probably when the window is created, it's state should be set to cause the client-side to load the "gclogo.ico" from the client-side p2j.jar.
- If a "window-icon" value is specified, this is essentially an application default icon that replaces the gclogo.ico as the default. This should invoke a **server-driven** icon loading process that is essentially the equivalent of a LOAD-ICON() method call.
- When the LOAD-ICON() or LOAD-SMALL-ICON() methods are called, they should do the following:
  1. try loading the image/icon from the **application** jar file
  2. if the resource was found in the jar, it should be loaded into memory and sent down to the client via a new ClientExports method named boolean setWindowIcon(int windowId, byte[] icon) (this avoids extra calls to the client), any failure will be returned by setWindowIcon() and should be returned from the LOAD-ICON() or LOAD-SMALL-ICON() method
  3. if not found in the application jar, use FileSystemOps.searchPath() to call down to the client to find the image/icon in the client-side filesystem using the propath
  4. if still not found in the file system return false
  5. else if found, then call a new ClientExports method named boolean loadWindowIcon(int windowId, boolean small, String name), any failure will be returned by loadWindowIcon() and should be returned from the LOAD-ICON() or LOAD-SMALL-ICON() method
- The LOAD-*IMAGE() variants should be reworked to operate in the same way, except they don't have any of the small-icon or default icon processing. But the same idea of being able to first load from the application jar and so forth... should be present in image loading too.

Notice that we don't have loading process invoked from the client side because that will require another round trip to the server to check the application jar. The design above takes advantage of server resources where possible and client resources otherwise and we try to minimize round trips.

4. Please check in the 4GL if the SEARCH() (propath searching) automatically gives the same current-directory results. I suspect that the 4GL propath already had the current directory and so it naturally worked. I'm worried that our search implementation is broken somehow because I'm surprised that we would have to explicitly add a current directory search.

#16 - 01/10/2015 05:21 PM - Eugenie Lyzenko

1. In ImageGuiImpl, should line 258 be if (readers.hasNext()) or should it be while (readers.hasNext())?

I use first available reader. This is the way recommended by TwelveMonkeys doc to handle files with multiple images inside.

Doesn't Progress return false if the icon is not found?

Yes, the Progress returns false in this case. P2J always returns true. This is because currently when server side calls icon loading we do not really load icon, just change window config and does not know if the file exists or not. And this is certainly mismatch we need to fix to reproduce P2J.

#17 - 01/12/2015 08:30 PM - Eugenie Lyzenko

The important finding. Progress does not load the icon from current directory if the current directory is not in PROPATH. Even SEARCH function returns ? in this case.

So we do not need to implement loading file from current directory if FileSystemOps.searchPath() does not resolve the filename to the fully qualified file system name.

#18 - 01/12/2015 08:41 PM - Greg Shah

OK, do whatever matches the behavior of the 4GL.

#19 - 01/14/2015 07:43 PM - Eugenie Lyzenko

- File evl_upd20150114a.zip added

The update includes icon related part of the icon/image implementation for review. Not completed testing but approach is here for you to know the way I take. So what is working:
- The icon is loading from server "window-icon"(server side) directory entry or gclogo.ico(client side) as default ones.
- The LOAD-*-ICON uses server to load icon from application JAR or client side to load icon from file system(if found in PROPATH).

- The LOAD-*-ICON now returns logical value depending on result.

Continue working.

#20 - 01/15/2015 11:33 AM - Greg Shah

Code Review evl_upd20150114a.zip

This is moving in the right direction, but I do have some questions/comments:

1. We want to minimize unnecessary calls from the client to the server. Generally, we want the server to dictate the icon for any new window, including the default window. In which scenarios does the WindowTitleBar constructor get invoked **before** the server has a chance to determine the default icon source? I would like to think that the server **always** can determine this and that we should have no need to use the getApplicationIcon() "up-call" to the server. Can we eliminate getApplicationIcon()?

It seems to me that **before** any window (even the default window) is created, the server should determine the default icon and send it down to the client. There are 3 possible types:

- customer-specific directory configured default icon that specifies an application jar resource (highest precedence, if this exists it is used)
- customer-specific directory configured default icon that specifies a client-side filesystem resource (the server can determine this when the filename cannot be loaded as a resource from the application jar), on the client side if this is found it is used
- gclogo.ico directly loaded by the client from the client-side p2j.jar (lowest precedence, if the first two cannot be found or are not specified then this is used and should always be there)

These three types can be specified with a default icon type enum ICON_CUSTOM_JAR_RESOURCE, ICON_CUSTOM_CLIENT_FILENAME, ICON_DEFAULT.

If ICON_CUSTOM_JAR_RESOURCE is specified, then there should also be a byte[] that was read from the application jar and sent down to the client.
If ICON_CUSTOM_CLIENT_FILENAME is specified, then there should be a filename String that is sent to the client.
If ICON_DEFAULT is specified, no other data is needed.

You can create a new ClientExport to setDefaultWindowIcon(typeenum, Object) and make sure that the server handles this for GUI before the default window is created. The resulting default icon instance can be cached in the client and reused.

For CREATE WINDOW, the ICON attribute is read-only right? So, one must always do the CREATE WINDOW first and then call LOAD-ICON() or LOAD-SMALL-ICON(). This means that for this case, we initially use the default icon that was previously setup for the client. Then we dynamically change the result using loadIcon() or loadSmallIcon() IF and when they are called.

In this case I see no need to ever "up-call" to the server for the icon.

2. I'm not sure what LogicalTerminal.loadImage(String, boolean) is supposed to do now. It should not be calling FileSystemOps.searchPath(). In the new design, this code should **first** check if the requested filename is a resource in the application jar. Then, if the name is not an application jar resource, then it must be a client side filesystem resource. But we don't want to call down to the client FileSystemOps.searchPath() which will just look for the file and then tell us the absolute path to the file on the client side. Because then we have to call down again to the client to tell it to actually load that file. We shouldn't be making 2 trips to the client for this when we can do it in a single trip.

3. LogicalTerminal.loadImage(String, boolean) has another (very major problem). It is using FileInputStream to load files from the filesystem on the server! But the filename that is being loaded is from FileSystemOps.searchPath() which is a client-side filename. I realize this was already in the code, but it is totally wrong. LT.loadImage() is server-side code and this can only ever work in the scenario where the clients and servers are running on the same system. But we don't want that code there at all. I think it was just a mistake.

#21 - 01/15/2015 12:31 PM - Eugenie Lyzenko

> 1. We want to minimize unnecessary calls from the client to the server. Generally, we want the server to dictate the icon for any new window, including the default window. In which scenarios does the WindowTitleBar constructor get invoked before the server has a chance to determine the default icon source? I would like to think that the server always can determine this and that we should have no need to use the getApplicationIcon() "up-call" to the server. Can we eliminate getApplicationIcon()?

This is the main question I thought about in icon implementation design. Yes, the server knows the default icon before the client window is on the screen. But we need to know the moment when we can push this icon down to the client(we can not set window icon if there is no window). So I thought the client can ask the server for icon in a appropriate time. But I'll think about to get rid of the getApplicationIcon().

> For 2 and 3 points.

In new design the method LogicalTerminal.loadImage(String, boolean) will be completely removed. I've left it to be able to test and debug intermediate results. In a today drop we do not need this method anymore.

#22 - 01/15/2015 12:53 PM - Greg Shah

Actually, let's name setDefaultWindowIcon() as setWindowDefaultIcon() instead. I think it is more descriptive.

> we can not set window icon if there is no window

Why not have ThinClient.setWindowDefaultIcon() call a static Window.setWindowDefaultIcon()? We have to save a single instance of it anyway and it will be shared for all windows. It seems to me that this can be done without any existing window instance.

#23 - 01/16/2015 02:05 PM - Eugenie Lyzenko

- File evl_upd20150116a.zip added

This update for review is not the final version to test. Just for you to see the default icon implementation and new image support. And not yet merged with the recent code base.

I need to perform code cleanup, moving common lines for separate method, example static Image.getData(InputStream), probably to another class(UiUtil is a good candidate).

But you can see the implementation details. Update merged with the recent code base will be uploaded later today.

#24 - 01/16/2015 02:27 PM - Greg Shah

Code Review evl_upd20150116a.zip

This is really good. Keep going as per your plan.

I realize this is unfinished. You may already know about these items, but just in case there are still some issues with:

1. Where FileSystemOps.searchPath() is called (on the server in ImageWidget.loadImageInt()).

2. Resource loading from the jar. Does LogicalTerminal.class.getResourceAsStream() load from the application jar file? It definitely will load from the p2j.jar, but the server side loading of an icon or image should not be looking in the p2j.jar.

3. Fallback to the default icon when other specified options fail.

#25 - 01/16/2015 03:50 PM - Eugenie Lyzenko

> 3. Fallback to the default icon when other specified options fail.

This is handled in LT.setWindowDefaultIcon():

```
...
         }
      }

      // if we are here - the last chance is to load embedded icon from the client side p2j.jar
      return locate().client.setWindowDefaultIcon(Window.IconDataType.ICON_DEFAULT, null);
   }
...
```

and this calls in Window:

```
...
   InputStream is = LogicalTerminal.class.getResourceAsStream(WindowConfig.DEFAULT_ICON);
...
```

on the client side. I guess this implements this feature.

#26 - 01/16/2015 03:52 PM - Greg Shah

Eugenie Lyzenko wrote:

> 3. Fallback to the default icon when other specified options fail.

> This is handled in LT.setWindowDefaultIcon():
> [...]

I think there is still a problem when the client side is told to do something (like load a filename) and that fails. At that point the client should handle the fallback automatically.

> and this calls in Window:
> [...]
> on the client side. I guess this implements this feature.

Usage of LogicalTerminal is a server-side thing, it can't work on the client.

#27 - 01/16/2015 04:13 PM - Eugenie Lyzenko

> I think there is still a problem when the client side is told to do something (like load a filename) and that fails. At that point the client should handle the fallback automatically.

OK. In Progress if application can not load the image with LOAD-*ICON - the window icon just does not change. For image case the Progress just loads nothing(with compiler or runtime messages).

#28 - 01/16/2015 04:54 PM - Greg Shah

I'm more worried about the client falling back to the default icon when the setWindowDefaultIcon() fails to load a filename or can't load the byte[] as an icon.

#29 - 01/16/2015 07:50 PM - Eugenie Lyzenko

- File evl_upd20150116b.zip added

This update has sources merged with the recent code base. Also cleanups, some common code rework(Image class removed from modifications). Still not sure if the resources are loaded correctly from application jar file. And client side now loads gclogo.ico if something is wrong on the client side during default icon creation(this is local call, not getting up to the server).

Continue working.

#30 - 01/17/2015 10:35 AM - Greg Shah

Code Review evl_upd20150116b.zip

1. You can't load LogicalTerminal on the client, but you are trying to do so. Window.setWindowDefaultIcon() uses LogicalTerminal.class.getResourceAsStream() which causes the LogicalTerminal.class to be loaded. This can have all kinds of unexpected bad behavior.

2. In Window.setWindowDefaultIcon(), when ICON_CUSTOM_CLIENT_FILENAME fails to find the matching file, why aren't you just falling through to the ICON_DEFAULT section? Instead you are relying upon extra processing in the WindowTitleBar constructor. That is not very clean because it requires the WindowTitleBar to know too much about the default icon processing. Instead, it should be able to simply read the default icon from the Window class.

3. WindowWidget.loadIconInt(), ButtonWidget.loadImageInt() and ImageWidget.loadImageInt(), LogicalTerminal.setWindowDefaultIcon() all have an incorrect design. Each one handles the full precedence hierarchy for icon/image loading, instead of letting the LogicalTerminal and ThinClient handle the parts more efficiently. For example, an icon/image loaded from the client filesystem requires 3 round trips to and from the client, but the approach described in note 20 above specifies that this should not happen. A method in LogicalTerminal should be called instead. That method will try to load first from the application jar. If that succeeds it will call the ThinClient.set*(byte[]) method. If the app jar load fails, then it calls the client with a filename. It is the ThinClient.set*(String filename) method which will handle BOTH the FileSystemOps.searchPath() AND the load*() WITHOUT RETURNING TO THE SERVER IN BETWEEN.

4. Code like the following is too verbose:

```
if (boolean_expression)
{
   return new logical(true);
}
else
{
   return new logical(false);
}
```

Please code this as:

```
return new logical(boolean_expression);
```

#31 - 01/19/2015 11:35 AM - Eugenie Lyzenko

One question.

> ...If the app jar load fails, then it calls the client with a filename. It is the ThinClient.set*(String filename) method which will handle BOTH the FileSystemOps.searchPath() AND the load*() WITHOUT RETURNING TO THE SERVER IN BETWEEN.

Does it mean the FileSystemOps.searchPath() must be called on client side? Because if we call it on the server side - there is no guarantee the file exists on the client side too, correct? So it is not correct to call FileSystemOps.searchPath() in LogicalTerminal, only in ThinCLient, right?

#32 - 01/19/2015 11:44 AM - Greg Shah

> Does it mean the FileSystemOps.searchPath() must be called on client side?

Not exactly. Instead, the client side must call FileSystemDaemon.searchPath().

> Because if we call it on the server side - there is no guarantee the file exists on the client side too, correct?

No. The FileSystemOps.searchPath() will call to the client and invoke FileSystemDaemon.searchPath(). So any call to FileSystemOps.searchPath() is expected to travel to the client and back. There is an exception for program names (.p files that are converted), but that doesn't come into play in your case.

> So it is not correct to call FileSystemOps.searchPath() in LogicalTerminal, only in ThinCLient, right?

Almost. As noted above you should call FileSystemDaemon.searchPath() from ThinClient.

#33 - 01/19/2015 02:28 PM - Eugenie Lyzenko

- File evl_upd20150119a.zip added

The new update for review. The fixes for notes to the previous one. Also the gclogo.ico default icon has been moved to the ui/client package to reflect the changes.

I've one one point to clarify. We can use client part of the FileSystemOps to search the file. But we need to use at least one call for FileSystemOps from the server side. Without this the PROPATH value on the client side is not initialized(is null) and we can not perform local file system search on the client PROPATH value. I have added the method init() to the FileSystemOps that does nothing except making such conversation. Yes, this is one more server->client call. But this is only once per session and without we do not have PROPATH on the client side.

#34 - 01/19/2015 05:43 PM - Eugenie Lyzenko

- File evl_upd20150119b.zip added

Small change to eliminate point #4 from the notes.

#35 - 01/19/2015 06:15 PM - Greg Shah

Code Review evl_upd20150119b.zip

This is really good!

The only thing I see is that FileSystemOps.searchPath() should not be called from the client side in Window.setWindowDefaultIcon().

#36 - 01/19/2015 06:53 PM - Eugenie Lyzenko

- File evl_upd20150119c.zip added

Greg Shah wrote:

> The only thing I see is that FileSystemOps.searchPath() should not be called from the client side in Window.setWindowDefaultIcon().

Yes, this code has serious bug in 0119b. Ironically I was planned to change is respectively. Anyway in this update the problem is fixed.

#37 - 01/19/2015 07:11 PM - Greg Shah

Code Review evl_upd20150119c.zip

I'm good with this.

Does this pass all of your 4GL testcases?

If so, please put it into conversion and runtime regression testing.

#38 - 01/19/2015 08:02 PM - Eugenie Lyzenko

I'm good with this.

Does this pass all of your 4GL testcases?

If so, please put it into conversion and runtime regression testing.

It's OK for default window loading. But I need to test all path/file combinations.

The button image widgets and DEFINE IMAGE support need more debugging/implementation.

May be we can finish icon support first, then continue with other image support.

#39 - 01/20/2015 10:42 AM - Eugenie Lyzenko

Regarding icon loading. Everything looks OK except one point I need to clarify:
1. Assume we start the server from /home/evl/root_dir/p2j/testcases/simple/server.
2. Assume this server startup directory has temp subdirectory and both ..../server and ..../server/temp directories has the icon files icon1.ico and icon2.ico.
3. When we call attempt to load icon from application jar on LT via:

```
...
      // check the application jar file first
      InputStream is = ClassLoader.getSystemClassLoader().getResourceAsStream(iconName);
...
```

if iconName is "icon1.ico" or iconName is "temp/icon2.ico" the is != null and presents the valid input stream to load data.

The question: Is it expected behavior for loading application resource on the server side? Why the server assumes it has such resource available for class loader? Yes, both icons are in the server loading directory, I just want to be 100% sure the server works as expected in this case.

If this point will be resolved, I suggest to run regression and commit icon related changes to have some intermediate work stage to shift the next works(button images and image widget itself). Is that OK?

#40 - 01/20/2015 10:56 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Regarding icon loading. Everything looks OK except one point I need to clarify:
> 1. Assume we start the server from /home/evl/root_dir/p2j/testcases/simple/server.
> 2. Assume this server startup directory has temp subdirectory and both ..../server and ..../server/temp directories has the icon files icon1.ico and icon2.ico.
> 3. When we call attempt to load icon from application jar on LT via:
> [...]
>
> if iconName is "icon1.ico" or iconName is "temp/icon2.ico" the is != null and presents the valid input stream to load data.
>
> The question: Is it expected behavior for loading application resource on the server side? Why the server assumes it has such resource available for class loader?

I can't tell you the exact algorithm getResourceAsStream is using, but what I recall is that is dependent on the order the resource folders/jars are added to the classpath. More, at some point I think it defaults to searching the OS file-system, even if only jars are in the classpath.

If you know the exact package of these files in the application jar, maybe we should consider appending the filename with the package name; if it's in the root package, add a "/" at the beginning.

#41 - 01/20/2015 10:59 AM - Greg Shah

It is not a bad idea to load resources from a specific part of the jar, but I am OK with the current implementation as a next step. No matter what we do, the filesystem will be searched for matching resources if the resource is not found in the jar. That is built into Java and we have no control over it without doing much more advanced things, which are not worth the effort.

#42 - 01/20/2015 10:59 AM - Greg Shah

> I suggest to run regression and commit icon related changes to have some intermediate work stage to shift the next works(button images and image widget itself). Is that OK?

To be clear, yes go ahead.

#43 - 01/20/2015 03:40 PM - Eugenie Lyzenko

The conversion testing completed. Generated codes are identical. Starting the runtime testing.

And I have committed image/icon testcases into bzr as 1254 for uast/image directory. The icon files are also there to play with different locations.

#44 - 01/21/2015 09:31 AM - Eugenie Lyzenko

The main part of the regression testing has been completed without regressions, starting the CTRL-C tests.

#45 - 01/21/2015 05:47 PM - Eugenie Lyzenko

The CTRL-C part completed without regressions. Had to start 3-way tests separately. The results file is on the shared drive: 10715_5c96d7c_20150121_evl.zip.

I'm going to commit and distribute the evl_upd20150119c.zip update.

#46 - 01/21/2015 06:03 PM - Greg Shah

Yes, go ahead.

#47 - 01/21/2015 06:17 PM - Eugenie Lyzenko

Committed in bzr as 10716.

#48 - 02/25/2015 09:07 AM - Greg Shah

- Status changed from New to Closed

#49 - 11/16/2016 12:13 PM - Greg Shah

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

#50 - 01/30/2017 01:28 PM - Greg Shah

- Related to Bug #2792: image loading issues added

Files

evl_upd20150107b.zip	20.1 KB	01/08/2015	Eugenie Lyzenko
image5-4gl.jpg	45.3 KB	01/08/2015	Eugenie Lyzenko
image5-p2j.jpg	190 KB	01/08/2015	Eugenie Lyzenko
evl_upd20150109a.zip	75.8 KB	01/09/2015	Eugenie Lyzenko
evl_upd20150109b.zip	75.9 KB	01/10/2015	Eugenie Lyzenko
evl_upd20150114a.zip	247 KB	01/15/2015	Eugenie Lyzenko
evl_upd20150116a.zip	300 KB	01/16/2015	Eugenie Lyzenko
evl_upd20150116b.zip	299 KB	01/17/2015	Eugenie Lyzenko
evl_upd20150119a.zip	318 KB	01/19/2015	Eugenie Lyzenko
evl_upd20150119b.zip	318 KB	01/19/2015	Eugenie Lyzenko
evl_upd20150119c.zip	317 KB	01/19/2015	Eugenie Lyzenko