User Interface - Feature #1795

implement printing support (real GUI printing not the child process stuff used in CHUI)

10/30/2012 10:01 AM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Hynek Cihlar	**% Done:**	100%
Category:		**Estimated time:**	120.00 hours
Target version:			
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to User Interface - Feature #3313: implement SYSTEM-DIALOG-PRINTER-SETUP	**Closed**
Related to Runtime Infrastructure - Bug #3338: performance of large report PD...	**New**

History

#1 - 10/31/2012 02:52 PM - Greg Shah

- Target version set to Milestone 12

#2 - 03/23/2016 04:54 PM - Greg Shah

- Target version deleted (Milestone 12)

#3 - 03/09/2017 05:08 PM - Greg Shah

Some useful references:

Can 4GL/ABL natively print images? (the short answer: NO)
Progress and Managing Font + Fint Size (the ABL can only do some rudimentary text output, with whatever default font is set at the time the graphics context is opened)
Controlling the Default Font for a Printer Context
Changing Font Size, Type and etc. while printing the document
"Manually Using WIN32 to Print (to avoid limitations in ABL):http://www.oehive.org/book/export/html/495

#4 - 03/09/2017 05:28 PM - Greg Shah

The following are the ABL features related to printing.

SYSTEM-DIALOG-PRINTER-SETUP
SESSION:PRINTER-NAME attr
SESSION:PRINTER-PORT attr
SESSION:PRINTER-CONTROL-HANDLE attr
SESSION:PRINTER-HDC attr (device context currently being used)
SESSION:GET-PRINTERS method
OUTPUT TO PRINTER

Read the docs for these carefully. The idea here is that images, printing of forms or any advanced graphical output is not done. Widgets don't render in any graphical manner. The printing support is really primitive. It is just text output to a graphics context that is created using:

- the system/user default
- the default after the user successfully makes changes during SYSTEM-DIALOG-PRINTER-SETUP
- overrides on the OUTPUT TO (for PRINTER-NAME, NUM-COPIES, LANDSCAPE, PORTRAIT, COLLATE)

Once the context is open, there is no modifying of these values. We have to figure out:

- when the device context is closed
- how to handle margins
- how to handle output that is too long (wrap it or truncate it)
- how the line spacing/positioning is done
- how any pagination is done (including the default settings, PAGED, PAGE-SIZE)

It is not clear exactly how STREAM-IO, CONVERT/NO-CONVERT/MAP, BINARY work. We need to experiment to see.

We can expect that the overall output would be done something like this (WIN32/GDI APIs here):

- figure out the printer context settings (GetDefaultPrinter et al)
- open the printer (OpenPrinter)
- get a device context (CreateDC)
- start a document (StartDoc)
- start a page (StartPage)
- output text content to the device context
- end a page (EndPage) and optionally start a new one...
- end document (EndDoc)
- delete the device context (DeleteDC)
- close printer (ClosePrinter)

We will want to implement a pure Java version of this, which normally would be pretty simple based on the Java2D printing support. The problem here is that we need to also support the web client.

I'm inclined to render a PDF for the web client and send it down as a URL that opens in a new tab. We want to maintain common code here, so it may mean that we have to provide driver-specific primitives for the replacement of the WIN32/GDI API calls. Then the code can be implemented at a higher level, with a driver-specific result.

#5 - 03/10/2017 03:19 PM - Greg Shah

Part of this solution is a replacement for the Progress adecomm/_osprint.p program. This outputs a text file to a printer. It takes parameters to control pagination, orientation and some other settings of the job.

In Linux/UNIX it reads a text file using IMPORT UNFORMATTED, and then uses DISPLAY to OUTPUT TO PRINTER with pagination. This is 4GL code.

On Windows, it calls ProPrintFile in the Progress PROPRINT.DLL. Supposedly, this brings up the printer setup dialog to allow the user to select the printer and control some job settings, then it outputs the text there. Since that code is native, we will have to experiment with the options to duplicate the result.

#6 - 07/26/2017 08:21 AM - Greg Shah

- Related to Feature #3313: implement SYSTEM-DIALOG-PRINTER-SETUP added

#7 - 07/26/2017 08:23 AM - Greg Shah

Please note that Ovidiu has handled the conversion parts of SYSTEM-DIALOG-PRINTER-SETUP in #3313 (see branch 1830a). The runtime aspects will need to be handled in this current task (#1795).

#8 - 08/11/2017 01:33 PM - Hynek Cihlar

- Assignee set to Hynek Cihlar

- Status changed from New to WIP

#9 - 08/11/2017 02:00 PM - Hynek Cihlar

OUTPUT TO PRINTER conversion support seems to be complete in current FWD trunk.

#10 - 08/11/2017 03:16 PM - Hynek Cihlar

I created task branch 1795a and checked in conversion support for SESSION attributes PRINTER-NAME, PRINTER-PORT, PRINTER-CONTROL-HANDLE, PRINTER-HDC and the method GET-PRINTERS.

This makes complete conversion support for all the printing language primitives. Please review.

#11 - 08/16/2017 10:02 AM - Greg Shah

Code Review Task Branch 1795a Revision 11158

The changes are good.

#12 - 08/17/2017 07:39 AM - Hynek Cihlar

1795a merged in to 3284a.

#13 - 10/04/2017 06:13 AM - Hynek Cihlar

In regards to the runtime support. OUTPUT TO PRINTER seems to behave the same as the other redirected outputs (file, terminal, etc.) except the output is sent to the configured printer device. So in order to make printing in GUI work we have to first make sure output redirection works in GUI, too. Is my assumption correct? I remember a while ago we had a discussion about output redirection in GUI with some ideas how to approach it. I think this is covered here #2368.

#14 - 10/04/2017 06:16 AM - Hynek Cihlar

Hynek Cihlar wrote:

> In regards to the runtime support. OUTPUT TO PRINTER seems to behave the same as the other redirected outputs (file, terminal, etc.) except the output is sent to the configured printer device. So in order to make printing in GUI work we have to first make sure output redirection works in GUI, too. Is my assumption correct? I remember a while ago we had a discussion about output redirection in GUI with some ideas how to approach it. I think this is covered here #2368.

...and this is covered by Constantin, I couldn't ask for more :-).

#15 - 10/04/2017 07:28 AM - Hynek Cihlar

SESSION:PRINTER-HDC returns the Win32 device context of the selected printer device. It can be used to render content directly to the printer using the Win32 API. This attribute will contain a meaningful value only when running (client?) on Windows and when printing to a physical printer (when implemented).

#16 - 10/04/2017 04:17 PM - Greg Shah

> So in order to make printing in GUI work we have to first make sure output redirection works in GUI, too. Is my assumption correct?

Maybe. But I am not sure. It really depends on what the 4GL does for the output. If I recall, it is doing a little bit more than just plain text output. It sets a font, uses a graphics context and may even handle some paging. If it is setting fonts etc... then it is not a ChUI output even if it is still just "slightly" formatted text.

> SESSION:PRINTER-HDC returns the Win32 device context of the selected printer device. It can be used to render content directly to the printer using the Win32 API. This attribute will contain a meaningful value only when running (client?) on Windows and when printing to a physical printer (when implemented).

Can you please look at the reports/application code to see what is exactly being done there. If it is used in a few enough places we will want to ask the customer if they are needed. And if they are needed, we will want to consider an alternative (like you did with the mouse pointer and screen refresh (which allowed us to get rid of the HWND usage).

#17 - 10/04/2017 04:43 PM - Hynek Cihlar

Greg Shah wrote:

> > So in order to make printing in GUI work we have to first make sure output redirection works in GUI, too. Is my assumption correct?
>
> Maybe. But I am not sure. It really depends on what the 4GL does for the output. If I recall, it is doing a little bit more than just plain text output. It sets a font, uses a graphics context and may even handle some paging. If it is setting fonts etc... then it is not a ChUI output even if it is still just "slightly" formatted text.

What I meant was that the character stream seems to be the same for file/terminal redirected output as well as the printer. And yes, more work must be done on top of the stream - fonts, maybe paging (although in case of paging it seems the PAGE-SIZE option doesn't care about the printer page size, it simply trims the content when the content doesn't fit on one printer page), etc.

#18 - 10/05/2017 10:05 AM - Greg Shah

Considering the memory problems with PDFBox, does it make sense to look at the jPod library instead?

#19 - 10/05/2017 11:38 AM - Hynek Cihlar

Greg Shah wrote:

> Considering the memory problems with PDFBox, does it make sense to look at the jPod library instead?

A potential problem with jPod is that it doesn't seem to be actively developed. The last commit is from December 2013. While PDFBox is backed up by the Apache organization, it is actively developed and has a large community. I'd like to test PDFBOx for the OUTPUT TO PRINTER case. If I find any memory issues I won't bother with it and switch to an alternative (jPod). The PDF rendering part of the Printer stream implementation is anyway just a thin layer with not much code that can be easily swapped for a different framework.

#20 - 10/05/2017 05:02 PM - Hynek Cihlar

1795b contains a very basic implementation of GUI printer stream that actually works with some hard coded assumptions (no printer management only OUTPUT TO PRINTER working, output to pdf file only, hard-coded page setup and pdf output file name). Since output redirection doesn't yet work in GUI, I set the GUI printer stream to be used in ChUI mode.

PDF rendering is done with PDFBox. And I am not sure what I did "wrong" but I didn't encounter any memory issues. I am able to generate pdfs with hundreds of thousands of pages without any issues or unusual memory profile. This is even without any attempt to optimize for memory, the output is kept in memory using the PDFBox'es PDDocument class and at the close of the stream it is saved to a pdf file.

#21 - 10/05/2017 05:14 PM - Sergey Ivanovskiy

- Related to Bug #3338: performance of large report PDF generation in admin UI is very slow added

#22 - 10/05/2017 05:27 PM - Hynek Cihlar

How far do we want to go when replicating the system printer dialog? With PDF file output as the only print output support at the moment can we reasonably simplify it? Also every Windows distribution has its unique appearance of the dialog. Since we support multiple Windows themes, do we want to also provide multiple versions of the dialog?

#23 - 10/05/2017 05:46 PM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> PDF rendering is done with PDFBox. And I am not sure what I did "wrong" but I didn't encounter any memory issues. I am able to generate pdfs with hundreds of thousands of pages without any issues or unusual memory profile. This is even without any attempt to optimize for memory, the output is kept in memory using the PDFBox'es PDDocument class and at the close of the stream it is saved to a pdf file.

Please share the code that generates non empty pdfs with hundreds of thousands of pages. It can help with 3338.

#24 - 10/05/2017 05:48 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Hynek Cihlar wrote:
>
> > PDF rendering is done with PDFBox. And I am not sure what I did "wrong" but I didn't encounter any memory issues. I am able to generate pdfs with hundreds of thousands of pages without any issues or unusual memory profile. This is even without any attempt to optimize for memory, the output is kept in memory using the PDFBox'es PDDocument class and at the close of the stream it is saved to a pdf file.
>
> Please share the code that generates non empty pdfs with hundreds of thousands of pages. It can help with 3338.

The Java implementation is in 1795b. The 4GL test case is below. Convert it and run it as usual in ChUI Swing client.

```
def var i as int.

output to printer page-size 10.

repeat i = 1 to 1000000:
   display "this is a test line" i.
end.
```

#25 - 10/05/2017 06:01 PM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Sergey Ivanovskiy wrote:
>
> > Hynek Cihlar wrote:
> >
> > > PDF rendering is done with PDFBox. And I am not sure what I did "wrong" but I didn't encounter any memory issues. I am able to generate pdfs with hundreds of thousands of pages without any issues or unusual memory profile. This is even without any attempt to optimize for memory, the output is kept in memory using the PDFBox'es PDDocument class and at the close of the stream it is saved to a pdf file.
> >
> > Please share the code that generates non empty pdfs with hundreds of thousands of pages. It can help with 3338.

> The Java implementation is in 1795b. The 4GL test case is below. Convert it and run it as usual in ChUI Swing client.

```
def var i as int.

output to printer page-size 10.

repeat i = 1 to 1000000:
   display "this is a test line" i.
end.
```

It seems the cool code. Can your code load approximately 3000 pages of the pdf report produced by P2J web admin. Using the documented code examples I couldn't load it in memory.

#26 - 10/05/2017 06:11 PM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Can your code load approximately 3000 pages of the pdf report produced by P2J web admin.

No, it can only render character stream.

I noticed one difference between the implementations (besides the fact that I only print text and no other graphic elements), I create only one PDPageContentStream per page, whereas in the Admin implementation there are many PDPageContentStream created for every page. I didn't study your code in more detail, but this is what I would try to address.

#27 - 10/05/2017 07:14 PM - Sergey Ivanovskiy

OK, understand, any advances can help here. The tested pdf report has 3661 pages and takes approximately 648 Mb of the hard disk space.

#28 - 10/06/2017 07:16 AM - Hynek Cihlar

Greg Shah wrote:

> Can you please look at the reports/application code to see what is exactly being done there. If it is used in a few enough places we will want to ask the customer if they are needed. And if they are needed, we will want to consider an alternative (like you did with the mouse pointer and screen refresh (which allowed us to get rid of the HWND usage).

I found one place where PRINTER-HDC is used. The value is only displayed to the user together with other session attributes.

#29 - 10/06/2017 07:33 AM - Greg Shah

OK, then just return 0xFFFFFFFF for it. -1 is an invalid HDC in Windows and it will be enough to satisfy the display. Most likely this is for development or support purposes anyway, since a real user would have no use for the HDC. Leave the assignment as an unimplemented feature.

#30 - 10/06/2017 08:38 AM - Greg Shah

The most common use of SYSTEM-DIALOG PRINTER-SETUP is:

```
procedure output-text-file-to-printer:
   define input parameter text-file-name as char.
   define input parameter is-sorted as logical.

   define variable prt-ok as logical no-undo.
   define variable prt-rc as logical no-undo.

   /* in some cases (but not all) there is this code, when this happens a 10 is passed as 3rd param to adecomm
/_osprint.p below */
   /*
      font-table:num-entries = 20.
      system-dialog font 10 fixed-only.
   */

   system-dialog printer-setup update prt-ok.
   if not prt-ok then return.

   /* 3rd param is sometimes set as 9 or 10 instead of 3; the following comment is in the code but */
   /* it is not always valid because there is no registry usage or SYSTEM-DIALOG FONT in most files: */
   /* Use Progress Print. Always use Font#9 in Registry (set above) */
   if is-sorted then
   do:
      /* 4th param as 0 means portrait */
      run 'adecomm/_osprint.p' (input ?, input text-file-name, input 3, input 0, input 0, input 0, output prt-
rc).
   end.
   else
   do:
      /* 4th param as 2 means landscape */
      run 'adecomm/_osprint.p' (input ?, input text-file-name, input 3, input 2, input 0, input 0, output prt-
rc).
   end.
end procedure.
```

FYI, there are other use cases for _osprint.p so we will have to consider them when writing that part.

This is an alternate use case for SYSTEM-DIALOG PRINTER-SETUP:

```
procedure alternate-output-text-file-to-printer:
   define input parameter text-file-name as char.

   define variable prt-ok as logical no-undo.
   define variable text-content as character format "x(176)" no-undo.

   system-dialog printer-setup update prt-ok.
   if not prt-ok then return.

   output to printer.
   /* there is also a case where it is a named stream used here (output stream rpt to printer page-size 78) an
d */
   /* then some following output to that named stream using PUT */

   input from value(text-file-name) no-echo.

   repeat:
      import unformatted text-content.
      if asc(substr(text-content, 1, 1) = 12 then
      do:
         page.
         text-content = substr(text-content, 2).
      end.
```

```
      if text-content ne "" then
         put unformatted text-content skip.
      else
         put unformatted skip(1).

      text-content = "".
   end.

   input close.
   output close.

end procedure.
```

My sense is that we can do a simplified implementation for now. Make sure that all the common options that a user would want are supported, but I think you don't have to implement a theme-specific match for each Windows-specific printer dialog.

I will check with the customer to confirm.

#31 - 10/19/2017 05:39 PM - Hynek Cihlar

It seems that the only motivation for using ProPrintFile (from PROPRINT.DLL) in _osprint.p is the ability to override the environment-configured printer font. ProPrintFile takes in one of its parameters a font number from the font table.

4GL code doesn't allow to override the printer font. So in order to fully replace ProPrintFile with a 4GL code the syntax of OUTPUT TO PRINTER should be extended.

#32 - 10/19/2017 06:17 PM - Greg Shah

> 4GL code doesn't allow to override the printer font. So in order to fully replace ProPrintFile with a 4GL code the syntax of OUTPUT TO PRINTER should be extended.

I don't have a problem with providing this, but for the current problem (replacing _osprint.p), I prefer to just provide the needed functionality in the runtime and we can convert RUN _osprint.p to a call to the new runtime method. The idea is that we don't strictly need to change the 4GL code for OUTPUT TO PRINTER.

Or were you planning to edit the src/adecomm/_osprint.p in possenet_v3.1_baseline_1_src_20040223_fixed_line_feeds.zip to use the 4GL extension and remove the native code?

#33 - 10/19/2017 06:19 PM - Greg Shah

To be clear, the version of _osprint.p that is to be used for any 4GL code editing must be the src/adecomm/_osprint.p in possenet_v3.1_baseline_1_src_20040223_fixed_line_feeds.zip because that is the version with the open source license.

#34 - 10/19/2017 06:28 PM - Hynek Cihlar

Greg Shah wrote:

> 4GL code doesn't allow to override the printer font. So in order to fully replace ProPrintFile with a 4GL code the syntax of OUTPUT TO PRINTER should be extended.

I don't have a problem with providing this, but for the current problem (replacing _osprint.p), I prefer to just provide the needed functionality in the runtime and we can convert RUN _osprint.p to a call to the new runtime method. The idea is that we don't strictly need to change the 4GL code for OUTPUT TO PRINTER.

Or were you planning to edit the src/adecomm/_osprint.p in possenet_v3.1_baseline_1_src_20040223_fixed_line_feeds.zip to use the 4GL extension and remove the native code?

Yes, this is what I did as this was my interpretation of the task assignment. I removed the native DLL call from _osprint.p and replaced it with the 4GL equivalent. The only missing feature is the ability to specify the print font. The original _osprint.p can stay as is and a new procedure file with the modified _osprint.p content can be introduced if needed.

But if for whatever reason you think we should go with the replacement of RUN _osprint.p at conversion, then I will introduce the new runtime method and modify conversion to call this new method.

#35 - 10/19/2017 06:31 PM - Greg Shah

> But if for whatever reason you think we should go with the replacement of RUN _osprint.p at conversion, then I will introduce the new runtime method and modify conversion to call this new method.

No, your approach is fine.

You are working with the possenet version?

#36 - 10/19/2017 06:36 PM - Hynek Cihlar

Greg Shah wrote:

> But if for whatever reason you think we should go with the replacement of RUN _osprint.p at conversion, then I will introduce the new runtime method and modify conversion to call this new method.

> No, your approach is fine.

Ok. I will add new option FONT fontNum to the OUTPUT TO PRINTER statement and use this in the modified _osprint.p.

> You are working with the possenet version?

That's correct.

#37 - 10/19/2017 06:36 PM - Greg Shah

Perfect.

#38 - 10/20/2017 04:16 PM - Hynek Cihlar

1795b rebased against trunk revision 11180.

#39 - 10/23/2017 04:01 AM - Hynek Cihlar

1795b rebased against trunk revision 11181.

#40 - 10/24/2017 09:26 AM - Greg Shah

Is this in a good place for a code review?

#41 - 10/24/2017 09:57 AM - Hynek Cihlar

Greg Shah wrote:

> Is this in a good place for a code review?

I am just putting in missing code comments, it will be ready shortly.

#42 - 10/24/2017 12:18 PM - Hynek Cihlar

1795b is ready for review.

The following are remaining tasks on the issue.

1. Finish regression testing.
2. Prettify the print setup dialog. I had some issues when building the dialog programmatically and postponed this till later. This will be a small change and localized only to the class that implements the dialog.
3. Resolve Utils.findDirectoryNodePath not finding the path when it should. According to the method comments the method should find the node in /server/default/runtime/default (among others), but it doesn't. I haven't spent much time on this yet, any ideas are appreciated.

#43 - 10/24/2017 04:32 PM - Hynek Cihlar

- File _osprint.p added

The attached file is the modified _osprint.p with the native code removed using the implemented print features in 1795b.

Please review.

#44 - 10/24/2017 07:13 PM - Hynek Cihlar

1795b revision 11193 resolves 1 from note 42 besides other fixes.

#45 - 10/25/2017 08:30 AM - Hynek Cihlar

I have checked in two regression fixes in 1795b revisions 11194 and 11195. I am restarting ChUI regression tests and continue with GUI regression testing.

#46 - 10/25/2017 11:04 AM - Hynek Cihlar

Another fix checked in to 1795b revision 11196. ChUI regression tests restarted.

#47 - 10/25/2017 02:12 PM - Hynek Cihlar

Rebased 1795b against trunk revision 11182.

#48 - 10/25/2017 04:34 PM - Hynek Cihlar

1795b 11200 passed GUI regression tests. ChUI still in progress, I got a technical error in previous run.

#49 - 10/25/2017 06:00 PM - Greg Shah

Code Review Task branch 1795b Revision 11200

Wow! This is really quite cool.

1. The thing I wonder about is how well the streaming works with large PDFs. Have you tested this with very large output?

2. I've checked in some changes to progress.g. These should be functionally the same as your previous approach. I also fixed ThinClient which had two history entries.

3. ModalWindow.setColorResolver() needs javadoc.

#50 - 10/26/2017 04:26 AM - Hynek Cihlar

Greg Shah wrote:

> Code Review Task branch 1795b Revision 11200
> 1. The thing I wonder about is how well the streaming works with large PDFs. Have you tested this with very large output?

I did several tests with outputs up to 500000 pages. I didn't find any unusual resource requirements and the PDF rendering performed well. This is also the reason I didn't attempt to do any optimizations, like rendering to temporary files, etc. Of course if we hit any specific work flows with special requirements we can address them later.

#51 - 10/26/2017 04:30 AM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Greg Shah wrote:
>
> > Code Review Task branch 1795b Revision 11200
> > 1. The thing I wonder about is how well the streaming works with large PDFs. Have you tested this with very large output?
>
> I did several tests with outputs up to 500000 pages. I didn't find any unusual resource requirements and the PDF rendering performed well. This is also the reason I didn't attempt to do any optimizations, like rendering to temporary files, etc. Of course if we hit any specific work flows with special requirements we can address them later.

Were these pdf pages in your test pure text pages without graphics?

#52 - 10/26/2017 04:32 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Were these pdf pages in your test pure text pages without graphics?

Yes, all were pure text. The redirected print output doesn't generate any graphic elements AFAIK.

#53 - 10/26/2017 09:33 AM - Hynek Cihlar

The latest 1795b resolves the points from review, point 3 from note 42, plus some related changes. I don't expect any other changes in the branch assuming the ChUI regression passes.

Please review.

#54 - 10/26/2017 09:46 AM - Greg Shah

Code Review Task branch 1795b Revision 11203

I'm good with the changes.

If it passes ChUI regression testing, please merge to trunk.

#55 - 10/28/2017 10:11 AM - Hynek Cihlar

1795b passed ChUI regression tests, was rebased against trunk revision 11183 and merged to trunk as revision 11184.

The task branch was archived.

#56 - 10/28/2017 10:11 AM - Hynek Cihlar

- % Done changed from 0 to 100

#57 - 10/28/2017 09:26 PM - Greg Shah

- Status changed from WIP to Closed

#58 - 01/17/2018 06:45 PM - Greg Shah

I've just tested the following program:

```
def var i as int.

message "Starting.".

output to printer page-size 10.

repeat i = 1 to 1000:
   display "this is a test line" i.
end.

output close.

message "Finished.".
```

I tried it in Swing ChUI, Swing GUI and Virtual Desktop GUI modes. I never see the PDF output... it runs without errors and no printer output results. Am I misunderstanding how this is supposed to work? I was expecting the PDF to be loaded into the user's browser for the web client. I'm not sure what was supposed to happen for the Swing clients.

#59 - 01/18/2018 04:30 AM - Hynek Cihlar

Greg Shah wrote:

> I tried it in Swing ChUI, Swing GUI and Virtual Desktop GUI modes. I never see the PDF output...

Greg, this seems to be a regression in trunk. Even file output doesn't work. I am looking into this.

#60 - 01/18/2018 08:43 AM - Hynek Cihlar

Hynek Cihlar wrote:

> Greg Shah wrote:
>
> > I tried it in Swing ChUI, Swing GUI and Virtual Desktop GUI modes. I never see the PDF output...
>
> Greg, this seems to be a regression in trunk. Even file output doesn't work. I am looking into this.

Please review a fix for this in 3435a/11255.

#61 - 01/18/2018 11:19 AM - Greg Shah

Code Review Task Branch 3435a Revision 11255

I'm good with the change.

1. Please review Open URL before answering the questions below.

2. In my testcase, the PDF is sent down to the browser in a manner that causes it to treat it like a download. The user is prompted to save it or open it with a local viewer. For OUTPUT TO PRINTER, I would prefer to load the PDF as a new tab in the browser. What changes do we need to implement that?

3. It would be useful to be able to load the PDF into a viewer from the Swing clients, otherwise OUTPUT TO PRINTER is broken there. What are your thoughts on doing this? It seems to me that P2J-OPEN-URL could be used for this.

4. We have some other requirements for P2J-OPEN-URL. This is being driven by the large GUI application we are working on right now). The requirements are not exclusively printing-related, but they will often be used to support various converted application output. Since the work is connected I'd like to deal with it at the same time as the other items above. The original work on P2J-OPEN-URL was done in #1815. Enhancements needed:

- Add the ability to load from 3 possible places:
  - A client-side resource, such as an application-generated output. The most common cases are .txt, .csv, .xls and .pdf. It could also just be a file that exists on the client.
  - A server-side resource, such as something from the application jar file.

    - A browser-accessible web resource via a proper URL. This could be on the Internet or the browser's internal network. This one is already working, though I think we might want to put some kind of security filtering in place.
- If the resource is coming from our server or client and it is not an HTML page, then I would expect the browser's application helpers may be required to deal with the resource being loaded. I think that means we may need to ensure that the MIME type is properly specified. For example, most 4GL GUI applications have quite a bit of direct integration with Excel. We would want to make it easy to send a spreadsheet down to the browser and let its configured application helper (maybe Excel) deal with the resource.
- Change the name of P2J-OPEN-URL to OPEN-URL since we are no longer putting a P2J prefix on our language extensions.

#62 - 01/18/2018 11:25 AM - Greg Shah

FYI, Stanislav is working to enhance the JasperReports support (see #3342) to handle XLS output. It already handles PDF quite nicely. In addition he is implementing export to PDF/XLS/CSV for enhanced browse (#3261) which will probably use the JasperReports support for its implementation. I expect that these will both be common sources of output that an application would need to load via OPEN-URL.

#63 - 01/18/2018 12:10 PM - Hynek Cihlar

Greg Shah wrote:

> Code Review Task Branch 3435a Revision 11255
>
> I'm good with the change.
>
> 2. In my testcase, the PDF is sent down to the browser in a manner that causes it to treat it like a download. The user is prompted to save it or open it with a local viewer. For OUTPUT TO PRINTER, I would prefer to load the PDF as a new tab in the browser. What changes do we need to implement that?

Currently the PDF is "opened" in the browser via window.open(pdf_url, _blank). The actual resulting behavior is browser and configuration dependent. For example when you install a PDF plugin in the browser, the pdf will open in its window, if not it will be downloaded to your local file system.

In order to guarantee the pdf to be opened in the browser (without forcing end-users into specific configurations) we would have to transform the pdf into DOM.

> 3. It would be useful to be able to load the PDF into a viewer from the Swing clients, otherwise OUTPUT TO PRINTER is broken there. What are your thoughts on doing this? It seems to me that P2J-OPEN-URL could be used for this.

Currently printed pdf outputs are stored in a predefined dir, by default it is the client's working dir.

Again, there is no guarantee a web browser would show the pdf directly, see my answer in point 2. Modern operation system GUI shells (Winodws, Linux, iOS, Android, etc.) however allow to "open" a URL. A system-registered handler that matches the specific URL would be launched given the URL to process. I.e. in case of a URL pointing to a PDF it would launch the default system pdf viewer (which may not be the browser). Of course this would apply only to the Swing client. For the web client the potential web browser limitations apply.

#64 - 01/18/2018 12:31 PM - Greg Shah

> For example when you install a PDF plugin in the browser, the pdf will open in its window, if not it will be downloaded to your local file system.

Considering that most browsers can already handle PDF natively (without any additional plugin), I think it may be better to implement a small html document that embeds the PDF and properly sets the mime type etc... This will allow the PDF to be rendered as a separate tab. The user can still save it locally, because the built-in browser PDF support enables it.

Text output might need a similar thing. If the browser doesn't load it by default, then we could wrap it in some simple html to render it. This would also allow the user to print the page using local printers. The problem there would be enabling local save.

CSV and XLS is probably OK the way it is with the user getting a dialog with "open using" or "save".

> A system-registered handler that matches the specific URL would be launched given the URL to process. I.e. in case of a URL pointing to a PDF it would launch the default system pdf viewer (which may not be the browser). Of course this would apply only to the Swing client. For the web client the potential web browser limitations apply.

P2J-OPEN-URL already handles the idea of knowing how to launch the browser (and allowing a custom override to be configured), across OS platforms, including both Web and Swing clients. Implementing the wrapper html idea from above should also work here I think.

#65 - 01/18/2018 03:28 PM - Hynek Cihlar

Greg Shah wrote:

> > For example when you install a PDF plugin in the browser, the pdf will open in its window, if not it will be downloaded to your local file system.
>
> Considering that most browsers can already handle PDF natively (without any additional plugin), I think it may be better to implement a small html document that embeds the PDF and properly sets the mime type etc... This will allow the PDF to be rendered as a separate tab. The user can still save it locally, because the built-in browser PDF support enables it.

For completeness. AFAIK Internet Explorer 11 doesn't handle pdfs natively and requires a plugin. Also depending on the browser settings embedding pdf with <embed> may not work. For example Chrome version 63 has an option "Download PDF files instead of automatically opening them in Chrome". When this is on, the <embed> tag won't show the pdf file (and won't even download it).

Also I can see that companies or individuals may prohibit to automatically open rich documents for security reasons.

However I agree that most mainstream browsers in their default settings will work.

#66 - 04/19/2019 01:51 PM - Greg Shah

Hynek Cihlar wrote:

> The attached file is the modified _osprint.p with the native code removed using the implemented print features in 1795b.
>
> Please review.

Is there any reason to NOT check this in to the bzr repo for possenet? In other words, is it a complete replacement for the original when used with FWD?

The only thing I can think of is that the code will only work in FWD now. We could use the prprocessor to ensure that our changes were only valid in FWD.

#67 - 04/23/2019 02:39 AM - Hynek Cihlar

Greg Shah wrote:

> Hynek Cihlar wrote:
>
> > The attached file is the modified _osprint.p with the native code removed using the implemented print features in 1795b.
> >
> > Please review.
>
> Is there any reason to NOT check this in to the bzr repo for possenet? In other words, is it a complete replacement for the original when used with FWD?
>
> The only thing I can think of is that the code will only work in FWD now. We could use the prprocessor to ensure that our changes were only valid in FWD.

The attached _osprint.p is already part of the Possenet repo. I added it to a Possenet zip file which later on became the initial revision of the repository.

The changes are functionally full replacement for the original version. The only think is that with the new FONT option in OUTPUT statement the modified _osprint.p won't run in OpenEdge.

#68 - 04/23/2019 07:34 AM - Greg Shah

I added it to a Possenet zip file which later on became the initial revision of the repository.

Ahh. That explains why I couldn't see the change in the revision history. Thanks!

Files

_osprint.p	8.5 KB	10/24/2017	Hynek Cihlar