Base Language - Feature #1631

Feature # 1623 (Closed): refine I/O support to resolve incompatibilities or missing features

implement support for LANDSCAPE/PORTRAIT in I/O statements

10/21/2012 10:37 AM - Greg Shah

Status:	Closed	**Start date:**	02/18/2013
Priority:	Normal	**Due date:**	09/13/2013
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	Cleanup and Stablization for Server Features		
billable:	No	**vendor_id:**	GCD

Description

Subtasks:

Feature # 2009: conversion support for LANDSCAPE/PORTRAIT I/O option	**Closed**
Feature # 2010: runtime support for LANDSCAPE/PORTRAIT I/O options	**Closed**

History

#1 - 10/31/2012 01:40 PM - Greg Shah

- Target version set to Milestone 7

#2 - 02/18/2013 07:07 PM - Greg Shah

The following is an example of the option in use:

```
OUTPUT STREAM s TO VALUE(my-filename-var) LANDSCAPE PAGE-SIZE 45.
```

PORTRAIT is used similarly. I believe they can appear in either the OUTPUT THROUGH or OUTPUT TO statements.

#3 - 02/19/2013 12:08 PM - Costin Savin

- File cs_upd20130219a.zip added

Added proposed update

#4 - 02/19/2013 03:34 PM - Greg Shah

Feedback:

Please fix the code formatting issues in Stream.java (there are several):

1. Header # is wrong.
2. Wrong spacing in header.
3. Missing blank line between methods.
4. Please don't use "@return See above.". Put a real description in there.

Otherwise it is OK.

#5 - 02/20/2013 03:04 AM - Costin Savin

- File cs_upd20130220a.zip added

Added proposed update.

#6 - 02/20/2013 08:28 AM - Greg Shah

It is good to go. I will let you know when we can schedule it. Because there are some updates in front, please be ready to merge it up to a later revision.

#7 - 02/22/2013 04:26 AM - Costin Savin

- File cs_upd20130222a.zip added

Added merged update

#8 - 02/22/2013 09:06 AM - Greg Shah

This is the wrong update (it is for the _MSG work). Please post the correct one.

#9 - 02/22/2013 09:07 AM - Costin Savin

- File cs_upd20130222b.zip added

Attached it

#10 - 02/22/2013 09:10 AM - Greg Shah

The update looks good. I will start testing it shortly. I am deleting the update that is not matching, so that there is no confusion in the future.

#11 - 02/22/2013 09:10 AM - Greg Shah

- File deleted (cs_upd20130222a.zip)

#12 - 02/22/2013 11:27 AM - Costin Savin

Conversion test passed, commited to bazaar as revision 10185

#13 - 04/25/2013 11:16 AM - Greg Shah

- Target version changed from Milestone 7 to Milestone 11

#14 - 05/02/2013 07:11 PM - Greg Shah

- Parent task set to #1623

#15 - 08/30/2013 09:46 AM - Greg Shah

The work above was all done to support the proper conversion of these options. But the runtime behavior has not yet been implemented. Please start by creating/testing enough standalone 4GL testcases to determine exactly the behavior of these options. Post the testcases here and check them into the testcases project. Then document the behavior that must be implemented.

A common mistake new P2J developers make is that they don't realize how much subtle behavior there is inside the 4GL. This leads to missing testcases and missing behavior in our implementation. To ensure you know the full behavior, you must make testcases that use the feature in all primary ways that the feature works, INCLUDING trying to break that feature by passing all possible variants of bad/unexpected input. Please review the testcases/uast/memptr/ testcases to get an idea of how extensive this may have to be.

#16 - 08/30/2013 04:31 PM - Vadim Gindin

I tried to run simple test with these options on lindev01 and windev01.
1. lindev01. I ran different test. See below.

```
OUTPUT TO somefile.data LANDSCAPE.
FOR EACH customer:
  DISPLAY cust-num name address SKIP(2) WITH 1 COLUMN SIDE-LABELS STREAM-IO.
END.
OUTPUT CLOSE.
DISPLAY "Finished".
```

This test had ran successfully with the same result for LANDSCAPE, PORTRAIT options or without them.
File had been created and options didn't affect direction of the text in it.

```
OUTPUT TO PRINTER LANDSCAPE.
FOR EACH customer:
  DISPLAY cust-num name address SKIP(2) WITH 1 COLUMN SIDE-LABELS STREAM-IO.
END.
OUTPUT CLOSE.
DISPLAY "Finished".
```

Program had been finished with error message: "lp - command not found", where lp - is a command wich outputs printers list.
And there is really no such command in lindev01. So in this case it is not possible to check if our options are working on lindev01.

2. windev01
I tried to run similiar script. But there was also no possibility to check if options are working because of access rights problem.\
There is on windev01 printer by default is a PdfCreator wich successfully called from script, but I can't save target file in my home folder.

#17 - 08/30/2013 05:57 PM - Greg Shah

> 1. lindev01. I ran different test. See below.

Please try both test programs on both lindev01 and windev01. In particular, the OUTPUT TO somefile.data LANDSCAPE approach will create a text file. Text files have no concept of portrait or landscape. So there may not be any different. However, it is possible that portrait or landscape may affect the default line length of each line of text. Please make sure that your DISPLAY line has wide enough content to test this idea.

> I tried to run similiar script. But there was also no possibility to check if options are working because of access rights problem.\

There is on windev01 printer by default is a PdfCreator wich successfully called from script, but I can't save target file in my home folder.

I will forward this to the customer.

#18 - 09/02/2013 10:07 AM - Vadim Gindin

I've made a script for testing conditions about width of text in the outut file. Here it is:

```
define var longstr as character format "x(150)" initial "012345678901234567890123456789012345678901234567890123456789012
345678901234567890123456789234" view-as text size 60 by 1.

output to longstr.dat .
display longstr with 1 column stream-io.
output close.

output to longstr_ls.dat landscape.
display longstr with 1 column stream-io.
output close.

output to longstr_prt.dat portrait.
display longstr with 1 column stream-io.
output close.
```

This script creates 3 files - for each variant: landscape(longstr_ls.dat), portrait (longstr_prt.dat), none of these options sepcified (longstr.dat). The content of that files are the same. It means that LANDSCAPE/PORTRAIT option do not affects the text width in text files.

#19 - 09/03/2013 10:48 AM - Greg Shah

Questions/Answers:

1. Why are you using the STREAM-IO in your tests? Unless it is needed, you generally should eliminate any extra options because they can effect/confuse your results.

2. The testcase in note 18 seems to limit the display width of the longstr variable to 60 characters since you are displaying it in a static text field that is only 60 chars wide. Why are you doing that? I think it may distort the results. Please note that it is rare to use a text widget. The default view-as is fill-in and that will display fine.

3. A better approach would be to simply display a long series of smaller values, in multiple rows and then use the landscape, portrait or no option to see the result. Something like this:

```
def var some-nums as int extent 20.
def var i as int.

do i = 1 to 20:
```

```
   some-nums[i] = i.
end.

output to some_nums_no_option.txt.
display some-nums.
output close.

output to some_nums_landscape.txt landscape.
display some-nums.
output close.

output to some_nums_portrait.txt portrait.
display some-nums.
output close.
```

Then run this in both Linux and Windows.

Then make sure to run it with different terminal widths (not just 80 chars, but much longer and shorter).

Then make a version that generates many rows. Pick a number (e.g. 200) that you are sure will be longer than the page length. Landscape or portrait modes may effect the default page size. Please review PAGED and PAGE-SIZE options and test with different combinations of these.

Review other options and see if you can find any that are possibly related. See if you can find any effects.

4. It is correct that we do not yet support CLIPBOARD or PRINTER as output destinations. This particular task should document the behavior of landscape and portrait options for BOTH the file output and the PRINTER output. If you cannot find any effect except with PRINTER, then we will defer further work until later (when we implement the PRINTER destination support).

5. On lindev01, please create a script named lp and put it in your path somewhere. In that script, see if you can just redirect the output into a file. This way you can test PRINTER on Linux. We need to know what to do for PRINTER support on Linux anyway, since it seems to actually work as a destination (even though the documentation states otherwise).

> Should I document the behavior of this options. You know that behavior is obvious: it only changes orientation of the page.

6. Yes, you should document all your findings (even if they are obvious) in how these options work on Windows with the PRINTER destination. As noted above, I think there is more testing to be done and potentially there are more things to document (e.g. page size, interactions with other options, no-effect on any destination except PRINTER...).

> Where to check in the testcases in the root folder "testcases" or to some subfolder?

7. Please use a sub-folder. There is already an testcases/uast/io/. You can put your files in a sub-folder there.

> The behavior of the OUTPUT TO is already implemented (PRINTER option too). Isn't it? Can I look at its implementation (source and runtime)?

8. I expect you to be reviewing the P2J source code for both the conversion (see rules/convert/input_output.rules) and the runtime (see src/com/goldencode/p2j/util/Stream*.java). If you need help finding where something is located, just ask.

#20 - 09/03/2013 04:25 PM - Vadim Gindin

The first test is for default terminal wich is xterm (80x24) on lindev01:

```
def var somenums as int extent 60.
def var i as int.

do i=1 to extent(somenums):
  somenums[i]=i.
end.

output to myterm_no_option.dat.
 display somenums no-labels format "999".
output close.

output to myterm_landscape.dat landscape.
 display somenums no-labels format "999".
output close.

output to myterm_portrait.dat portrait.
 display somenums no-labels format "999".
output close.
```

It creates 3 identical files. So LANDSCAPE/PORTRAIT options don't affect page size in this case.

The second test is for terminal wy370-132 (width=132):

```
def var somenums as int extent 60.
def var i as int.
def var oldterm as char.

do i=1 to extent(somenums):
  somenums[i]=i.
end.

oldterm = TERMINAL.
TERMINAL="wy370-132".

output to myterm_wy370-132_no_option.dat.
 display somenums no-labels format "999" with width 132.
output close.

output to myterm_wy370-132_landscape.dat landscape.
 display somenums no-labels format "999" with width 132.
output close.

output to myterm_wy370-132_portrait.dat portrait.
 display somenums no-labels format "999" with width 132.
output close.

TERMINAL=oldterm.
```

It creates 3 identical files with textwidth=132 symbols. So in this case our options also do not affect the page width.

#21 - 09/03/2013 04:41 PM - Greg Shah

It creates 3 identical files with textwidth=132 symbols. So in this case our options also do not affect the page width.

You will have to test with wider than 132 and narrower than 132 terminals to confirm that the width 132 controls the output.

#22 - 09/04/2013 04:32 PM - Vadim Gindin

1. While starting progress in terminal width = 60. Message: "Unable to use your terminal. Check your PROTERMCAP file. (443)
The size of the variable terminal window should have a minimum of 73 columns and a minimum number of 7 rows. (4146)
You might want to make your terminal larger. (4147)". So I couldn't shrink the terminal to sizes less then specified in the message.
2. If I set 7 rows as terminal height - Progress couldn't open menu because of its size. So I increased rows size to 15 rows.
3. I ran the first test for terminal 73x15. No differencies. (LANDSCAPE, PORTRAIT, no option)
4. Attributes paged and page-size. Default page-size is 56. LANDSCAPE, PORTRAIT options do not affect page size - default or specified using page-size option.

```
&GLOBAL-DEFINE LEN 10
def var somenums as int extent {&LEN}.
def var i as int.
def var j as int.

run init_somenums.
output to myterm_no_option_paged.dat paged page-size 30.
repeat i=1 to 200:
  display somenums no-labels format "9999".
  do j=1 to {&LEN}:
    somenums[j] = somenums[j] + {&LEN}.
  end.
end.
output close.

run init_somenums.
output to myterm_portrait_paged.dat portrait paged page-size 30.
repeat i=1 to 200:
  display somenums no-labels format "9999".
  do j=1 to {&LEN}:
    somenums[j] = somenums[j] + {&LEN}.
  end.
end.
output close.

run init_somenums.
output to myterm_landscape_paged.dat landscape paged page-size 30.
repeat i=1 to 200:
  display somenums no-labels format "9999".
  do j=1 to {&LEN}:
    somenums[j] = somenums[j] + {&LEN}.
  end.
end.
output close.

/* init array with nums from 1 to LEN */
procedure init_somenums:
  do i=1 to extent(somenums):
    somenums[i]=i.
  end.
end procedure.
```

This test creates multipaged files with symbol ^L delimiting pages. These files are identical for options and page size is correct.
5. MAP option of OUTPUT TO statement do not affect width or height of content. I ran previous test with parameter "map wy370-132" but without any effect on width or height of content.

6. Talking about lp. It is a command wich sends files to printer. This command is not installed on lindev01 (if I understant this situation correctly). I created new folder and added it to PATH. After that I created a executable file lp with a one string: echo "It works!". Going further I ran simple script from Progress:

```
OUTPUT TO PRINTER.
DISPLAY "Some message".
OUTPUT CLOSE.
```

This script called my file lp and message "It works!" appeared in console. The problem is that I don't understand the mechanism of transferring the file to lp command. I echoed the parameters wich is sent by progress to lp command. There was only one "-s" wich means silent mode - without showing ID of created processes. I suspect that this command is configured to transfer data from stdin to printer.
But at this comment I don't know how the Progress transfers data to lp command and how to redirect it to file.

The other possible way is to set up pdf printer on lindev01. In my laptop it was only one package needed to install: "cups-pdf". Is it possible to install this package on lindev01 within the lp command?

#23 - 09/04/2013 04:53 PM - Greg Shah

> 4. Attributes paged and page-size. Default page-size is 56. LANDSCAPE, PORTRAIT options do not affect page size - default or specified using page-size option.

The testcase as shown tests with paged page-size 30 which is equivalent to just using page-size 30. Did you separately test with the following?

- neither paged nor page-size
- paged by itself
- page-size larger than 56

> This script called my file lp and message "It works!" appeared in console. The problem is that I don't understand the mechanism of transferring the file to lp command. I echoed the parameters wich is sent by progress to lp command. There was only one "-s" wich means silent mode - without showing ID of created processes.

I think creating an lp.sh with this content will probably do the job:

```
#!/bin/bash

echo "$0 Parameters: " "$@" > lp_parameters.txt
cat > lp_output.txt 2>&1
```

Anything passed as command line arguments will be output to lp_parameters.txt in the current dir.

Anything written to the stdin will be output to lp_output.txt in the current dir.

> I suspect that this command is configured to transfer data from stdin to printer. But at this comment I don't know how the Progress transfers data to lp command and how to redirect it to file.

I'm sure that Progress just launches a child process for lp and writes a stream of the output to the lp process' stdin.

> The other possible way is to set up pdf printer on lindev01. In my laptop it was only one package needed to install: "cups-pdf". Is it possible to install this package on lindev01 within the lp command?

Let's try it with the script first. The cups stuff may need a graphical environment, which is not available. Even if it doesn't it is a more complicated solution.

#24 - 09/04/2013 04:56 PM - Greg Shah

> 1. Greg, you wroted in the task 1631: "You will have to test with wider than 132 and narrower than 132 terminals to confirm that the width 132 controls the output." What do you mean by your last phrase: "the width 132 controls the output"?

I mean for you to confirm if the use of the 4GL option "width 132" in the 4GL frame phrase will cause the redirected file output to have a maximum of 132 columns regardless of the width of the terminal.

#25 - 09/05/2013 03:36 PM - Vadim Gindin

The next script is:

```
&GLOBAL-DEFINE LEN 10

def var somenums as int extent {&LEN}.
def var i as int.
def var j as int.
repeat i=1 to {&LEN}:
  somenums[i]=i.
end.

output to printer paged page-size 60.
  repeat j=1 to 200:
    display somenums no-labels format "9999".
    repeat i=1 to {&LEN}:
        somenums[i] = somenums[i] + 10.
    end.
  end.
output close.
```

1) OUTPUT TO PRINTER on lindev01 works through lp with bash script, you adviced. After it executes the lp_output.txt contains the output of the procedure. So it works and I ran some test cases using this command. One thing.. The Progress documentation says that landscape/portrait options should be supported by printer driver, i.e. real behaviour depends on printer driver. Should we try to install cup-pdf or at this moment it is enough to use this "ghost" of lp command?

2) page-size cases, greater than default 56. Output here is a line number of page breaks (^L symbol) in output file lp_output.txt for each option below:

no options: ***57,113,169,205***
paged: ***57,113,169,205***
page-size 60: ***61,121,181,205***
paged page-size 60: ***61,121,181,205***
landscape: ***57,113,169,205***
landscape paged: ***57,113,169,205***
landscape page-size 60: ***61,121,181,205***
landscape paged page-size 60: ***61,121,181***
portrait: ***57,113,169***
portrait paged: ***57,113,169***
portrait page-size 60: ***61,121,181***
portrait paged page-size 60: ***61,121,181***

When page-size is specified explicitly (by value 60), consequenced numbers of lines, containing a page break symbol(^L) are 61, 121 (61+60), 181 (121+60) as expected. Landscape/portrait options don't affect the output content.

When page-size is not specified explicitly (the default value 56 is used), sequental numbers of lines, containing a page break symbol(^L) are 57, 113 (57+56), 169 (113+56) as expected. Landscape/portrait options don't affect the output content.

3) case of frame phrase with width 132. I ran the second test for this case for the terminal with widths 80 (<132) and 150 (>132). The result is that output width is mininimal number from terminal width and width, specified in frame phrase. 132 in our case. landscape portrait options are also don't affect the output.

#26 - 09/05/2013 04:53 PM - Greg Shah

> The Progress documentation says that landscape/portrait options should be supported by printer driver, i.e. real behaviour depends on printer driver. Should we try to install cup-pdf or at this moment it is enough to use this "ghost" of lp command?

The key questions here are:

1. What are the parameters passed to lp by Progress for these different cases (landscape, portrait, none)?
2. In each of these cases (landscape, portrait, none), is the lp_output.txt identical?

The reason I ask is that if the "printer driver" is responsible for this handling landscape/portrait AND the generated data stream is identical, then the only way for a Linux "printer driver" to be involved in this is by Progress changing the parameters to lp. If Progress is not changing the parameters to lp (and the lp_output.txt is identical in all cases), then that means that Linux tolerates the existence of landscape/portrait option BUT DOESN'T ACTUALLY HONOR IT AT RUNTIME.

We shouldn't need to install any cups printer to know the answer.

#27 - 09/06/2013 07:16 AM - Vadim Gindin

Greg Shah wrote:

> The key questions here are:
>
> 1. What are the parameters passed to lp by Progress for these different cases (landscape, portrait, none)?

Always one parameter: "-s"

> 1. In each of these cases (landscape, portrait, none), is the lp_output.txt identical?

Yes, lp_output.txt is identical for landscape/portrait options with the same page-size.

#28 - 09/06/2013 07:24 AM - Greg Shah

Interesting. OK, this means that on Linux the options are silently ignored. Our runtime implementation will have to do the same thing.

Finish your testing on Windows and document your results. Then we can make a final plan.

Last thing: your results also now tell us how to implement the PRINTER destination support for Linux. I will assign that task to you. You can include that change in your changes for this task.

#29 - 09/06/2013 11:08 AM - Vadim Gindin

- File output_to_printer_simple_row.p added

- File output_to_printer_landscape_paged_150.pdf added

- File output_to_printer_landscape_paged_60.pdf added

I ran all tests on windev01 and made additional test. The common results are:

1. LANDSCAPE/PORTRAIT options are not affected by page-size options.
2. In the case of writing the output to a file LANDSCAPE/PORTRAIT options do not affect the content as it is in a Linux system.
3. When the page-size (specified explicitly) is greater then real height of a page, the content crosses bottom border of the page and become invisible. (see output_to_printer_landscape_paged_150.pdf). As I understand this behaviour is correct.
4. There is some strange effects, no linked with LANDSCAPE/PORTRAIT options. The content width on a page (lets call it a PAGEWIDTH) is calculated implicitly and differently than it goes on Linux. When the line width > PAGEWIDTH, then new line (or lines) allocated for the line remaining (see output_to_printer_landscape_paged_60.pdf also). It seems, that default PAGEWIDTH is 80 and Progress reserves last several symbols for the some special case (see example with formatting error).
5. Another strange is how the Progress writes separates fields. See the script below.

```
&GLOBAL-DEFINE LEN 9

def var somenums as int extent {&LEN}.
```

```
def var i as int.
repeat i=1 to {&LEN}:
  somenums[i]=1.
end.

output to printer "PDFCreator".
display somenums NO-LABELS.
output close.
```

> This script is just print simple array of 9 symbols "1". I expected that the result pdf will content one line. But the actual result really surprised me (see output_to_printer_simple_row.p). LANDSCAPE/PORTRAIT are also don't affect the result of this script in any way.

As a result it seems that the only functionality of this options is an orientation of the page regardless of content as expected. Isn't it?

#30 - 09/06/2013 11:45 AM - Greg Shah

- Status changed from New to WIP

> I expected that the result pdf will content one line. But the actual result really surprised me (see output_to_printer_simple_row.p). LANDSCAPE/PORTRAIT are also don't affect the result of this script in any way.

Please post the resulting PDF here so I can look at it.

> As a result it seems that the only functionality of this options is an orientation of the page regardless of content as expected. Isn't it?

Yes.

For now, I just want you to handle the cases of using LANDSCAPE/PORTRAIT with:

- OUTPUT TO file
- OUTPUT TO PRINTER on Linux

We will leave the OUTPUT TO PRINTER on Windows until later (when we add printing support to the GUI).

#31 - 09/06/2013 12:27 PM - Vadim Gindin

- File output_to_printer_simple_row.pdf added

Here is the resulting file for surprising test.

#32 - 09/06/2013 12:35 PM - Greg Shah

The UI layout manager will do this when the columns won't fit into the width available. It is actually quite a complicated set of layout algorithms with many moving parts.

#33 - 09/06/2013 01:16 PM - Vadim Gindin

I reviewed the input-output rules and the code in classes /goldencode/p2j/util/Stream*.java
I'm going to do the folowing things:

1. add new method for the StreamFactory called openPrinterStream and implement it for Linux (calling lp command and ignoring LANDSCAPE/PORTRAIT options). The LANDSCAPE/PORTRAIT options are supported by Stream.displayMode.
2. add rules to input-output.rules wich uses new method and pass to it these options.
   After that it should work and I'll be able to run and test it. Do I understand correctly?

#34 - 09/06/2013 02:39 PM - Greg Shah

Your plan is almost exactly right.

> add new method for the StreamFactory called openPrinterStream

Yes.

> implement it for Linux (calling lp command and ignoring LANDSCAPE/PORTRAIT options)

Yes. You may have to implement a PrinterStream subclass, that knows how to handle the platform-specific details. Be aware that the code for this class really runs on the P2J client side and that the StreamFactory itself resides on the P2J server. There is a "remoting" of the stream that is being used here.

In addition, the child process support is already implemented in ProcessStream, so you should use that for the actual implementation.

> add rules to input-output.rules wich uses new method

Yes.

> and pass to it these options.

No. input_output.rules already has this code:

```
      <rule>parent.type == prog.io_options

         <rule>type  == prog.kw_portrait
            <action>
                refid = createJavaAst(java.method_call,
                                      "setPortrait",
                                      parentid)
            </action>
         </rule>
```

```
        <rule>type  == prog.kw_landscap
           <action>
               refid = createJavaAst(java.method_call,
                                     "setLandscape",
                                     parentid)
           </action>
        </rule>
```

This means that AFTER the stream is created and returned by the StreamFactory, the options are set into it by the converted application code. You don't have to worry about passing the options to the StreamFactory.openPrintStream() method.

> After that it should work and I'll be able to run and test it.

Yes.

I think that the normal FileStream implementation already (by default) ignores the LANDSCAPE/PORTRAIT options, so there is probably nothing to do for that case.

#35 - 09/09/2013 02:49 PM - Vadim Gindin

I need some help.

1. Rules.
I've made a rule which converts OUTPUT TO PRINTER to this code:

```
UnnamedStreams.assingOut(StreamFactory.createPrinterStream(), true, false);
```

I don't understand how second and third arguments are generated. What rules are responsible for that?
Is it correct, that in this case UnnamedStreams is used?

2. Implementation.
Monday is a hard day..) Actually I'm in a little frustration about implementation of the PrinterStream class. There are two ways of implementation as I think:
a) try to manually execute "lp -s" command
StreamFactory creates a PrinterStream for us. I just don't understand how it should work. This PrinterStream is assigned for the program as output stream. When should I ran the lp command with this PrinterStream as input stream? The only place is the end of the converted program. Therefore it seems I should write some post-rules which will generate code, which will ran lp command. I stuck...
b) or try to use some j2se classes for it (platform independent). This impl may looks like this:

```
    public void print(InputStream source) {
        PrintService printService = PrintServiceLookup.lookupDefaultPrintService();
        DocPrintJob job = printService.createPrintJob();
        DocFlavor docFlavor = DocFlavor.INPUT_STREAM.AUTOSENSE; //"octet/stream" MIME type
```

```
        DocAttributeSet docAttributes = new HashDocAttributeSet();
        docAttributes.add(OrientationRequested.PORTRAIT); // substitute landscape/portrait options here
        Doc doc = new SimpleDoc(source, docFlavor, docAttributes);
        try {
            job.print(doc, null);
        } catch (PrintException e) {
            // TODO implement exception handling
        }
    }
```

#36 - 09/09/2013 03:16 PM - Greg Shah

I've made a rule which converts OUTPUT TO PRINTER to this code:

UnnamedStreams.assingOut(StreamFactory.createPrinterStream(), true, false);

I don't understand how second and third arguments are generated. What rules are responsible for that?

I recommend a search for "java.bool_" in the input_output.rules file. Either java.bool_true or java.bool_false is needed to generate a boolean literal. I haven't looked through all of them, but this one looks like it might cause one of the cases:

```
      <rule>
         type == prog.kw_to            and
         parent.type == prog.output_to and
         !descendant(prog.kw_term, 1)
         <action>
            createJavaAst(java.bool_true, "", closestPeerId)
         </action>
      </rule>
```

You should also search for all the prog.output_to cases to see what logic is getting executed. Remember that you are adding a new OUTPUT TO case that previously did not exist, so the code generation rules need to be more specific (to differentiate the PRINTER case from the general file case).

Is it correct, that in this case UnnamedStreams is used?

Yes, since you used OUTPUT TO PRINTER, there is no named stream in this case.

If you used OUTPUT stream my-stream-name TO PRINTER, then the result would be different. Please try both to understand the difference. In fact, you should make sure you have a testcase that tests this path as well.

a) try to manually execute "lp -s" command

Not exactly, but this is close.

b) or try to use some j2se classes for it (platform independent). This impl may looks like this:

No, definitely don't do this. The result in the 4GL is to REALLY invoke "lp -s". Whatever we implement MUST do this exactly.

> StreamFactory creates a PrinterStream for us. I just don't understand how it should work. This PrinterStream is assigned for the program as output stream. When should I ran the lp command with this PrinterStream as input stream? The only place is the end of the converted program. Therefore it seems I should write some post-rules which will generate code, which will ran lp command. I stuck...

We already have support for "process streams":

```
def var txt as char init "This is some bogus data.".

/* open unnamed output process stream */
output through cat > some_output_file.txt.

/* write the text to the process stream */
display txt.

/* close unnamed output process stream */
output close.
```

Convert this code and examine it. Look through the classes that are used and how the child process gets launched. Run the code and debug through it to understand how it works.

Then consider that we should re-use the process stream support for the Linux PRINTER case. Of course, the stream that is used should be selected at runtime and any use of "lp -s" should be hidden in our runtime and not in the generated code. That way, we can add a Windows implementation that is also hidden inside the runtime and the same generated code will do different things depending on the runtime platform (exactly like in the 4GL).

#37 - 09/11/2013 11:33 AM - Vadim Gindin

I didn't find the reason to write a separate stream class such PrinterStream, but I find the other way. We can add new static method launchPrinterJob to ProcessOps and substitute its call through corresponding rules. I think that from the process launching point of view there is no differencies in comparing to launching usual remote process like it goes in example procedure that you wroted. The ProcessOps accumulates methods of process launching so it would be essential to add new method there. Here is the content of proposed method:

```
   public static void launchPrinterJob(RemoteStream sin) {
```

```
        // a process launch always sets the last error to 0
        FileSystemOps.setLastError(FileSystem.ERR_NO_ERROR);

        if(sin == null)
            throw new IllegalArgumentException("sin is not specified");

        String[] cmdlist = new String[] { "lp", "-s" }; /
        work.obtain().rl.launch(cmdlist, -1, sin.getId());
    }
```

What do you think?

#38 - 09/11/2013 11:49 AM - Greg Shah

I don't want to mix Linux PRINTER output device support in with generic process launching. In addition, Windows PRINTER output device support will need something completely different (not process launch based), so putting the code in ProcessOps would make the generated code Linux-specific which is something we cannot do.

To cleanly hide the implementation of PRINTER output device support, I want to hide these things inside a PrinterStream class.

Of course, that class should re-use the process launching stream facilities when on Linux/UNIX. For now, it needs no implementation for Windows. We will add that later.

#39 - 09/11/2013 01:47 PM - Vadim Gindin

As I understand StreamDaemon and ProcessDaemon are executed on the Client (on the same node). How can I obtain ProcessDaemon from StreamDaemon directly? I've created the implementation of StreamDaemon.openPrinterStream() and it needs to call ProcessDaemon.launch method. Here is the code:

```
StreamDaemon:
    @Override
    public int openPrinterStream() {
        int streamId;
       if(!PlatformHelper.isUnderWindowsFamily() /* TODO make more appropriate check for linux*/) {
          ProcessStream printerStream = new ProcessStream();
          streamId = store(printerStream);

          /* it's new static method. This method is not neccessary and the static call is not good here,
             so I would better get ProcessDaemon instance here and call its existing launch method instead
          */
          ProcessDaemon.launch(streamId, printerStream);
       } else { /* windows */
          Stream printerStream = new NullStream();
          streamId = store(printerStream);
          // TODO implement further logic
       }
       return streamId;
    }
```

#40 - 09/11/2013 02:10 PM - Greg Shah

The ProcessDaemon actually already has a reference to StreamDaemon. When we instantiate both in ThinClient, we already have a dependency between the 2 classes. Just add StreamDaemon.registerProcessDaemon(ProcessDaemon) instance method and then call it from inside the ProcessDaemon constructor.

In regard to your code, please be advised that it does not match our coding standards. You should be familiar with them before you submit the code for official review.

#41 - 09/12/2013 02:29 PM - Vadim Gindin

During the testing step I faced with a problem. When a printer name is specified in the progress OUTPUT TO PRINTER statement, the conversion process makes wrong Java statement.

For the Progress statement:
OUTPUT TO PRINTER "somePrinter"
There will be Java statement:
UnnamedStreams.assignOut(StreamFactory.openPrinterStream(), "\"somePrinter\"");
It leads to compile error.

In the common case it would be better to pass printer name as a parameter of openPrinterStream method, in spite of the fact that printer name is ignored in linux and default printer is used.

Anyway, I have a several problems here:
1) It seems, that printer name conversion is not described in input_output.rules and goes as standard literal conversion. At least I could not find the corresponding place in that file. Could you help me to locate the target place in rules?
2) I found in .parser file that printer name was parsed as FILENAME. I expected that it will be parsed as SYMBOL. Is it correct? If not how to correct it?
3) In resulting Java statement there are extra quotes. How to prevent this effect?
4) Is there a way to override standard processing of this literal?
5) It seems I should add the parameter "printer name" to openPrinterStream() method. What to pass to it at this moment - real literal or null for simplicity?

#42 - 09/12/2013 02:56 PM - Greg Shah

> OUTPUT TO PRINTER "somePrinter"

Interesting. Please note that the Progress 4GL Language reference states this:

```
PRINTER [ printer-name ]

By default, this option sends output to the printer defined in the default print
context. Specify a printer name to send output to a specific printer. Specifying a
printer name overrides, but does not change, the printer defined in the default print
```

```
context.

When you use this option, it implies that the device you are sending output to is
paged, unless you also specify PAGE-SIZE 0.

In Windows, you must specify network printers in Universal Naming Convention
format. For example:

\\fs_dev\hplas4

On UNIX, the printer spooling facilities (lp or lpr) are used automatically.
```

Notice that they do not describe the printer name as a "character" type. Their example even suggests it is NOT a string literal.

So when I implemented the parser's rule to match this syntax, I implemented it as this (see com/goldencode/p2j/uast/progress.g):

```
/**
 * Matches the PRINTER keyword with its optional following
 * printer name as matched by {@link #filename}.
 * <p>
 * This rule is called by {@link #io_from_to_stmt}.
 */
printer_clause
   :
      KW_PRINTER^
      (
           // avoid collisions with the following options
           {
              LA != KW_BINARY   && LA != KW_ECHO     &&
              LA != KW_NO_ECHO  && LA != KW_UNBUF    &&
              LA != KW_MAP      && LA != KW_NO_MAP   &&
              LA != KW_NO_CVT   && LA != KW_CONVERT  &&
              LA != KW_NUM_COPY && LA != KW_PAGE_SZ  &&
              LA != KW_COLLATE  && LA != KW_LANDSCAP &&
              LA != KW_PORTRAIT && LA != KW_APPEND   &&
              LA != KW_PAGED    && LA != KW_KEEP_MSG
           }?
           filename
         | // the empty alternative
      )
   ;
```

Please look at progress.g and the printer_clause and filename rules to understand what that means.

Some questions:

1. Does the 4GL accept input for the printer name that is NOT a string literal? In many places in the 4GL, you can specify a filename right in the code instead of as a string literal (which would appear to the parser as a STRING node).
2. Does the 4GL actually work properly when you specify the printer name as a string literal?
3. Are you sure that the printer name is ignored on Linux/UNIX? Could they pass that to lp as a parameter?

Based on this, we (actually, you) will have some changes to make in the parser (progress.g).

> In the common case it would be better to pass printer name as a parameter of openPrinterStream method, in spite of the fact that printer name is ignored in linux and default printer is used.

Yes, this is necessary since the same 4GL code can be used on non-Linux platforms and it must still work. Please do document (in PrinterStream) the facts of how it behaves differently for Linux/UNIX and Windows.

> 1) It seems, that printer name conversion is not described in input_output.rules and goes as standard literal conversion. At least I could not find the corresponding place in that file. Could you help me to locate the target place in rules?

We won't know if this is an issue or not until you answer the questions above.

2) I found in .parser file that printer name was parsed as FILENAME. I expected that it will be parsed as SYMBOL. Is it correct? If not how to correct it?
3) In resulting Java statement there are extra quotes. How to prevent this effect?
4) Is there a way to override standard processing of this literal?

All of these are consequences of parsing this phrase using the filename rule in the parser. Depending on what you find for the questions above, we may or may not need to fix this in rules, but probably not. The bottom line is that if the parser generates a STRING node, that node will naturally convert to a Java string literal with no extra effort. Likewise, a "real" FILENAME node (that is not actually a string literal) will safely convert to a Java string literal. So it is likely that if the parser is fixed, the rest comes "for free".

5) It seems I should add the parameter "printer name" to openPrinterStream() method. What to pass to it at this moment - real literal or null for simplicity?

Don't change the openPrinterStream(), just ADD a variant which is openPrinterStream(String). That way, the common case (where there is no printer name) is handled cleanly. But the alternate is there to handle the explicit name case.

Finally, note this comment from the 4GL docs: On UNIX, the printer spooling facilities (lp or lpr) are used automatically.. Please try to determine when lpr is chosen in preference to lp, since we will have to duplicate that too (if the docs are right).

#43 - 09/12/2013 04:34 PM - Vadim Gindin

It seems I really missed this circumstance. See the results of my testing below.

Some questions:

1. Does the 4GL accept input for the printer name that is NOT a string literal? In many places in the 4GL, you can specify a filename right in the code instead of as a string literal (which would appear to the parser as a STRING node).

Yes, both in Linux and Windows

1. Does the 4GL actually work properly when you specify the printer name as a string literal?

It works properly on Windows but doesn't work on Linux. If printer name is specified as string literal - it don't passed to lp command.

1. Are you sure that the printer name is ignored on Linux/UNIX? Could they pass that to lp as a parameter?

You are right. If printer name is specified without quotes - it passed to lp command with -d parameter, but with peculiar properties:

```
OUTPUT TO PRINTER somePrinter     => lp -s -d somePrinter
OUTPUT TO PRINTER \\Comp\Printer  => lp -s -d CompPrinter
OUTPUT TO PRINTER \\Comp\\Printer => lp -s -d CompPrinter
OUTPUT TO PRINTER \\Comp/Printer  => lp -s -d Comp/Printer
```

As we can see the symbol '\' is lost in parameter of lp command.

#44 - 09/12/2013 04:56 PM - Vadim Gindin

> Finally, note this comment from the 4GL docs: On UNIX, the printer spooling facilities (lp or lpr) are used automatically.. Please try to determine when lpr is chosen in preference to lp, since we will have to duplicate that too (if the docs are right).

I couldn't determine when lpr is chosen. lp was always chosen. I found the difference between these command is that lp is a front end to off-line command lpr (see [[http://wiki.answers.com/Q/What_is_the_conceptual_difference_between_the_lp_and_lpr_commands_in_Unix]]). So I suspect that 4GL is allways use lp.

#45 - 09/12/2013 05:28 PM - Greg Shah

OK, interesting results. The progress.g will need to be fixed and most everything else from a conversion perspective will come "for free".

The only problem I see is a runtime issue: since we emit both a FILENAME and a STRING as a Java String literal, we will have to somehow "save" the information of whether the input was originally a filename or a string since Progress differentiates these cases at runtime. This is just one more reason to hate the 4GL. Crazy.

#46 - 09/13/2013 08:26 AM - Vadim Gindin

Could you tell me please how to specify the rules to convert OUTPUT TO PRINTER printerName statement to correct Java statement and pass the printerName to createPrinterStream method? I couldn't find how to do it in input_output.rules.

#47 - 09/13/2013 08:28 AM - Greg Shah

Your question is too broad. Are you asking for information on how to code TRPL programs? Do you have some specific problem you are trying to solve that you can't figure out?

#48 - 09/13/2013 09:09 AM - Vadim Gindin

I'll explain.
Let's assume, that we are converting the statement OUTPUT TO PRINTER "printerName". At this moment result of conversion is:
UnnamedStreams.assignOut(StreamFactory.openPrinterStream(), "somePrinter"); But I need that result:
UnnamedStreams.assignOut(StreamFactory.openPrinterStream("somePrinter")); Talking about jast, that is a result of 4GL ast conversion, there is a new node automatically appears in jast for the string "printerName". Here the jast for overall statement:

```
<ast col="0" id="266287972387" line="0" text="UnnamedStreams.assignOut" type="STATIC_METHOD_CALL">
  <annotation datatype="java.lang.Long" key="peerid" value="261993005059"/>
  <ast col="0" id="266287972388" line="0" text="StreamFactory.openPrinterStream" type="STATIC_METHOD_CALL"/>
  <ast col="0" id="266287972389" line="0" text="\"somePrinter\"" type="STRING">
    <annotation datatype="java.lang.Long" key="peerid" value="261993005066"/>
  </ast>
</ast>
```

As we can see the target ast[id="266287972389"] appears in the wrong place of tree. I need it appears as a child node of ast[id="266287972388"].

So, I tried to add a corresponding rule to the input-output.rules file. But as a result I got an additional ast as a child of ast[id="266287972389"]. I suspect that the string literal conversion rules are desribed not in input_rules.rules. I need to override/change the conversion logic to put the target ast in the right place in the jast. That is the question: how to do it.

#49 - 09/13/2013 09:44 AM - Greg Shah

You are on the right track by looking at the tree itself to understand what is generated. But the problem here is likely associated with the input tree (the Progress ast). What does this snippet of the tree look like (please show the entire subtree for the OUTPUT TO PRINTER "something" code)?

> I suspect that the string literal conversion rules are desribed not in input_rules.rules.

Of course. Strings are used everywhere and have nothing specifically to do with I/O. We convert them generically. You can find that code in literals.rules.

Your problem is not with how the string emits. It is with the input tree.

#50 - 09/13/2013 10:03 AM - Vadim Gindin

Here is 4GL tree part corresponding to that statement:

```
<ast col="0" id="261993005058" line="0" text="statement" type="STATEMENT">
  <ast col="0" id="261993005059" line="0" text="i/o stmt" type="OUTPUT_TO">
    <ast col="8" id="261993005060" line="1" text="to" type="KW_TO">
      <ast col="11" id="261993005063" line="1" text="printer" type="KW_PRINTER">
        <ast col="19" id="261993005066" line="1" text=""somePrinter"" type="FILENAME"/>
      </ast>
    </ast>
  </ast>
</ast>
```

Don't see at the type FILENAME here. After I'll fix the parser the target ast[id="261993005066"] will have the type="STRING".

#51 - 09/13/2013 10:14 AM - Greg Shah

> Don't see at the type FILENAME here.

What do you mean by this? The FILENAME node is there.

> After I'll fix the parser the target ast[id="261993005066"] will have the type="STRING".

Yes, you will for the string case. But the unquoted text should still be parsed as FILENAME (the way it is today).

The tree structure here should naturally help you emit everything in the right place. But for this to work, make sure you are creating the JAST for the KW_PRINTER as a PEER node. Please paste your TRPL rule here for creating that node. I suspect that is the issue.

#52 - 09/13/2013 10:24 AM - Vadim Gindin

Greg Shah wrote:

> Don't see at the type FILENAME here.

> What do you mean by this? The FILENAME node is there.

Sorry, I only ment that I want to consider only the string case here. Never mind. Lets consider both cases.

> After I'll fix the parser the target ast[id="261993005066"] will have the type="STRING".

> Yes, you will for the string case. But the unquoted text should still be parsed as FILENAME (the way it is today).

I understand. Sure.

> The tree structure here should naturally help you emit everything in the right place. But for this to work, make sure you are creating the JAST for the KW_PRINTER as a PEER node. Please paste your TRPL rule here for creating that node. I suspect that is the issue.

Here is my rule for the FILENAME type:

```
<rule> type ==prog.filename and parent.type == prog.kw_printer
   <action>
      createPeerAst(java.null_literal, "", closestPeerId)
   </action>
</rule>
```

By the way, two questions:
#What is the difference between PeerAst and JavaAst
#What is closesPeerId (depends on first question)

Thanks

#53 - 09/13/2013 11:48 AM - Greg Shah

> Here is my rule for the FILENAME type:

Actually, I don't think this code is needed. The FILENAME node should convert naturally. This code is part of the problem.

Please show the code for creating the openPrinterStream() JAST node.

> What is the difference between PeerAst and JavaAst
> What is closesPeerId (depends on first question)

There is not really an AST that is the "PeerAst". In other words, we only deal with 2 kinds of AST today: ProgressAst and JavaAst. These are real Java classes that represent tree nodes that have language-specific token types.

Of course, when we generate JAST (JavaAst instances) from AST (ProgressAst) nodes, typically a great deal of the structure of the JAST tree is directly built of the same AST structure that already exists. Of course, there are places where the generated JAST subtree deviates from the original structure, but this is less common.

When we create a "peer" JAST from a Progress AST, we store a "peerid" annotation in the "source" Progress AST that has as its value the JAST ID. We similarly store the Progress AST's ID in the JAST as its "peerid" annotation. Since ID values are unique across the entire project, each 64-bit ID uniquely identifies a specific AST/JAST node. This means that if you have a reference to the JAST, you can easily get its "peer" node which is the Progress AST node. And vice versa, if you have the Progress AST node, you can get its peer node that is the JAST. The linkage is bidirectional.

This is useful for many things, but it is especially useful as a "trick" automatically determining the JAST parent node to which to attach a new JAST child. This is where the closestPeerId comes in. When you reference closestPeerId, TRPL returns back the "peerid" annotation for the nearest parent node (or grandparent...) that has a "peerid" annotation. Often this is just the parent, but sometimes the peerid is read from higher in the tree. Remember that this peerid references the JAST peer of the parent node, so by default attaching a new JAST child node to the closetPeerId will make it "naturally" emit in the same structure as the original Progress tree.

#54 - 09/13/2013 01:26 PM - Vadim Gindin

Here is the rule, that creates openPrinterStream()

```
<rule>descendant(prog.kw_term, 1)
    <!-- pass null to the assignIn() or assignOut() if we are referencing the terminal -->
    <action>
        createPeerAst(java.null_literal, "", closestPeerId)
    </action>
    <rule on="false">this.getChildAt(0).type == prog.kw_os_dir and parent.type == prog.input_from
          <action>
                createPeerAst(java.static_method_call,
                            "StreamFactory.openDirStream",
                            closestPeerId)
          </action>

          <!-- common case: a file is being referenced, construct it -->
         <rule on="false">descendant(prog.kw_printer, 1) and parent.type == prog.output_to

          <action>
                  createJavaAst(java.static_method_call,
                               "StreamFactory.openPrinterStream",
                                #(long) copy.parent.getAnnotation("peerid"))
          </action>
          <action on="false">
                  createPeerAst(java.static_method_call,
                                "StreamFactory.openFileStream",
                                closestPeerId)
          </action>
          </rule>
       </rule>
   </rule>
</rule>
```

#55 - 09/13/2013 01:45 PM - Greg Shah

OK, there are multiple things to discuss here.

1. First, the code you have posted is nested inside this "top level" rule:

```
      <!-- create the file stream (the filename will automatically emit) -->
      <rule>
         (type == prog.kw_from and parent.type == prog.input_from) or
         (type == prog.kw_to   and parent.type == prog.output_to)
```

Any code inside this will execute with the current node ("this" or "copy") will be either KW_FROM or KW_TO. Since the PRINTER case can only occur with OUTPUT TO, your code should always execute with KW_TO as "this".

Then you added this:

```
         <rule on="false">descendant(prog.kw_printer, 1) and parent.type == prog.output_to

          <action>
                  createJavaAst(java.static_method_call,
                                "StreamFactory.openPrinterStream",
                                 #(long) copy.parent.getAnnotation("peerid"))
          </action>
```

By using createJavaAst(), you are creating a non-peer JAST. The JAST will not have any peerid annotation. And the KW_TO node will not have a peerid annotation either. If you change it to createPeerAst(), the KW_TO will have the peerid annotation and all the options will emit naturally into that method call as parameters.

This code #(long) copy.parent.getAnnotation("peerid") should be the equivalent to closestPeerId. Please use closestPeerId in preference.

#56 - 09/17/2013 03:45 AM - Vadim Gindin

At this moment these statements can correctly be converted:

output to printer
output to printer printerName
output to printer "printerName"

Faced with a problems with converting options.

output to printer paged page-size 60
UnnamedStreams.assignOut(StreamFactory.openPrinterStream("page"), 60, true)

output to printer printerName paged page-size 60
UnnamedStreams.assignOut(StreamFactory.openPrinterStream("printerName"), 60, true)

My rule:

```
<rule>type == prog.kw_printer and parent.type == prog.kw_to
  <action>
    createPeerAst(java.static_method_call,
                  "StreamFactory.openPrinterStream",
                  closestPeerId)
  </action>
</rule>
```

Here is .ast:

```
<ast col="0" id="304942678018" line="0" text="statement" type="STATEMENT">
  <ast col="0" id="304942678019" line="0" text="i/o stmt" type="OUTPUT_TO">
    <annotation datatype="java.lang.Long" key="peerid" value="309237645347"/>
      <ast col="8" id="304942678020" line="1" text="to" type="KW_TO">
        <ast col="11" id="304942678023" line="1" text="printer" type="KW_PRINTER">
          <annotation datatype="java.lang.Long" key="peerid" value="309237645348"/>
            <ast col="19" id="304942678026" line="1" text=""somePrinter"" type="STRING">
              <annotation datatype="java.lang.Long" key="peerid" value="309237645349"/>
            </ast>
        </ast>
      </ast>
      <ast col="0" id="304942678029" line="0" text="options" type="IO_OPTIONS">
        <ast col="33" hidden="true" id="304942678030" line="1" text="paged" type="KW_PAGED"/>
           <ast col="39" id="304942678033" line="1" text="page-size" type="KW_PAGE_SZ">
              <ast col="49" id="304942678035" line="1" text="60" type="NUM_LITERAL">
                <annotation datatype="java.lang.Boolean" key="use64bit" value="false"/>
              </ast>
            </ast>
        </ast>
      </ast>
    </ast>
  </ast>
</ast>
```

The rule uses a parentid variable as a parent node pointer(as other IO_OPTIONS do) and this var is correctly set as OUTPUT TO statement As I understand the corresponding rules - it should work, But it don't.
Here is .jast:

```
<ast col="0" id="309237645347" line="0" text="UnnamedStreams.assignOut" type="STATIC_METHOD_CALL">
  <annotation datatype="java.lang.Long" key="peerid" value="304942678019"/>
    <ast col="0" id="309237645348" line="0" text="StreamFactory.openPrinterStream" type="STATIC_METHOD_CALL">

      <annotation datatype="java.lang.Long" key="peerid" value="304942678023"/>
        <ast col="0" id="309237645349" line="0" text="somePrinter" type="STRING">
          <annotation datatype="java.lang.Long" key="peerid" value="304942678026"/>
        </ast>
    </ast>
    <ast col="0" id="309237645350" line="0" text="60" type="NUM_LITERAL"/>
    <ast col="0" id="309237645351" line="0" text="" type="BOOL_TRUE"/>
</ast>
```

#I don't understand why the parentid var value is changed during execution?
#We emit openPrinterStream node as peer node with correspondant annotation, so any other nodes wich use closesPeerId will find it as a parent. We must stop this logic after IO_OPTIONs started.. How can we do this?

#57 - 09/17/2013 12:21 PM - Greg Shah

> The rule uses a parentid variable as a parent node pointer(as other IO_OPTIONS do) and this var is correctly set as OUTPUT TO statement As I understand the corresponding rules - it should work, But it don't.

In the original rules, parentid is assigned in only 1 place:

```
      <!-- convert INPUT FROM, OUTPUT TO or I/O THRU into the proper stream
           variable reference, instantiation and assignment -->
      <rule>
         type == prog.input_from  or
         type == prog.output_to   or
         evalLib("io_thru_stmt", this)

         <!-- save off the parent for where options will be attached (this
              works because these statements can't be nested within
              one another) -->
         <action>parentid = closestPeerId</action>
```

Please note that this assignment is BEFORE the section that actually creates AST nodes that correspond to the OUTPUT_TO node. This means that we are saving off the peerid of the PARENT of the OUTPUT_TO node. This is some kind of BLOCK node and it is a "trick" for making sure that we emit the options code as separate lines in the same block where the OUTPUT_TO will emit.

```
output to printer paged page-size 60
UnnamedStreams.assignOut(StreamFactory.openPrinterStream("page"), 60, true)

output to printer printerName paged page-size 60
UnnamedStreams.assignOut(StreamFactory.openPrinterStream("printerName"), 60, true)
```

Please see the usage of the variable setPage to understand how the page size options emit.

> We emit openPrinterStream node as peer node with correspondant annotation, so any other nodes wich use closesPeerId will find it as a parent. We must stop this logic after IO_OPTIONs started.. How can we do this?

closestPeerId will only read the peerid from nodes that are direct ancestors. Because of this, closestPeerId only affects child nodes. It cannot affect the child nodes of siblings (nieces or nephews). Please look at CommonAstSupport for the implementation code.

#58 - 09/17/2013 06:54 PM - Vadim Gindin

- File vig_20130918a.zip added

Here is an archive with my changes. Please review. What to do further?

#59 - 09/18/2013 12:14 PM - Greg Shah

Code Review 0918a

This is pretty good work. Some changes to make:

1. The name of the update should be vig_upd20130918a.zip.

2. Import statements should not be defined class-by-class. For example, in StreamDaemon, this:

```
import com.goldencode.util.PlatformHelper;
```

should be this:

```
import com.goldencode.util.*;
```

3. The new StreamDaemon.processDaemon member is missing javadoc.

4. Why is the command line for the child process defined as "lps"? I thought the program was "lp".

5. I have found a Progress 4GL command line parameter that is related to your work:

```
-o printername
```

The description from the Progress docs:

> Identifies the printer to use when processing OUTPUT TO PRINTER statements

More interestingly, I have found an example of real usage in a customer application's .pf file (which is a way to pass a standard list of preferences to Progress without having to specify each one on the command line):

```
-o          "np -s -q hpcorp3"  # Output to printer
```

This suggests that you can override the entire default printer command line string, including custom parameters!

Instead of command line parms, we handle this by allowing (optional) entries to be added to the directory. Please see date.initWindowingYear() for an example of how to implement this feature. In other words, you would define the default string (as a private static final String member of StreamDaemon. Then you would pass this as the default value to the DirectoryManager.getString() method. The result will have to be parsed before usage. The result will allow P2J deployments to customize the printer string on a per-account, per-group, per-server or installation-wide basis.

Please test this command line parameter to confirm my suspicions. Then implement the feature as documented here.

6. There should be a space between the if and open (in if(in StreamDaemon.

7. Do you really need to do printerName.replaceAll("\\", "") in StreamDaemon? Or would this happen automatically by the Linux/UNIX shell?

8. Please put the code for generating the child process command line into its own method. Use constants and data that has been pre-processed as much as possible to reduce the number of places command line text is hard coded.

9. Please don't insert extra blank lines at the end of the file (before the closing curly brace for the class). See StreamDaemon.

10. Please don't remove blank lines that have been inserted (potentially) for formatting purposes. For example, line 925 of input_output.rules.

11. This code in progress.g is not needed above the STRING alternative:

```
            // avoid collisions with the following options
            {
               LA(1) != KW_BINARY   && LA(1) != KW_ECHO     &&
               LA(1) != KW_NO_ECHO  && LA(1) != KW_UNBUF    &&
               LA(1) != KW_MAP      && LA(1) != KW_NO_MAP   &&
               LA(1) != KW_NO_CVT   && LA(1) != KW_CONVERT  &&
               LA(1) != KW_NUM_COPY && LA(1) != KW_PAGE_SZ  &&
               LA(1) != KW_COLLATE  && LA(1) != KW_LANDSCAP &&
               LA(1) != KW_PORTRAIT && LA(1) != KW_APPEND   &&
               LA(1) != KW_PAGED    && LA(1) != KW_KEEP_MSG
            }?
```

The reason for the code in the filename case is that filename can match keywords, including that specific list of keywords that can be referenced in OUTPUT TO PRINTER. But a STRING token cannot be confused with these keywords and so it doesn't need this semantic predicate in order to work properly.

Once these changes are made, post a new update zip and I will do the final review. Of course, make sure your testcases all work first.

#60 - 09/20/2013 04:21 PM - Vadim Gindin

- File vig_upd20130918b.zip added

I'm uploading next update. I made described corrections.
About slashes, it seems that Progress removes it itself. I stayed this code.

#61 - 09/20/2013 05:12 PM - Greg Shah

Code Review 0918b

1. The name of this update should have been vig_upd20130920a.zip since you completed your changes on 0920 and it is the first update of the day.

2. All changes are good. The only thing I see is that there is too much printer-specific logic in StreamDaemon. I should have noticed this before, but it is more clear now that there are many printer-specific data members. This is why I originally suggested a PrinterStream class. I would like you to implement a PrinterStream class. For now, the class doesn't have to be a subclass of Stream. Instead, just move your current logic into a static factory method in PrinterStream, which can return a Stream instance. For Linux, the instance will be a ProcessStream. And for now on Windows, the NullStream is fine. Eventually, the class will be made into a Stream subclass and on Windows an instance of itself will be returned. But that is not needed now.

After you have that reworked, upload the code for review. And you can go ahead and run both conversion and runtime regression testing on devsrv01. I will do the review as soon as possible over the weekend, but this way you can make headway in testing before I get the final review done.

#62 - 09/23/2013 09:22 AM - Vadim Gindin

- File vig_upd20130923a.zip added

Next update

#63 - 09/23/2013 11:15 AM - Greg Shah

Code Review 0923a

1. There is no PrinterStream class in this update. I think you accidentally included ProcessStream (which is not changed).

2. The StreamDaemon.openPrinterStream() still has too much knowledge of the internals of the printer implementation. Please implement this more like:

```
public int openPrinterStream(String printerName)
{
   return store(PrinterStream.createPrinterStream(printerName));
}
```

All the setup logic for the printer can be hidden inside PrinterStream, right? This results in cleaner code and we only have one place to look for logic related to the printer implementation.

#64 - 09/23/2013 12:18 PM - Vadim Gindin

Greg Shah wrote:

> Code Review 0923a
>
> 1. There is no PrinterStream class in this update. I think you accidentally included ProcessStream (which is not changed).

You're right. Sorry.

> 2. The StreamDaemon.openPrinterStream() still has too much knowledge of the internals of the printer implementation. Please implement this more like:
>
> [...]
>
> All the setup logic for the printer can be hidden inside PrinterStream, right? This results in cleaner code and we only have one place to look for logic related to the printer implementation.

Actually, the only instantiation can be hidden inside PrinterStream. It because I have to initiate processLaunching after StreamDaemon.store() using StreamDaemon.processDaemon. Even If I'll move processDaemon to PrinterStream it uses streamId as a result of StreamDaemon.store() anyway.

#65 - 09/23/2013 12:24 PM - Greg Shah

Then just do the call to SD.store() from inside PrinterStream.createPrinterStream(). Hiding the printer-specific stuff is more important than limiting the reverse-dependency. Move the registerProcessDaemon() to PrinterStream and make it static. Then call there to register the ProcessDaemon reference.

#66 - 09/23/2013 01:26 PM - Vadim Gindin

- File vig_upd20130923b.zip added

corrected archive

#67 - 09/23/2013 03:37 PM - Greg Shah

Code Review 0923b

OK. Functionally the code is good. The last issues are about the code formatting/naming...

1. Please don't use a qualified reference to the PrinterStream class for static member access within the current class. So use processDaemon instead of PrinterStream.processDaemon and generateLinuxPrinterCommand() instead of PrinterStream.generateLinuxPrinterCommand(). There is no ambiguity in the references so the extra text should be avoided.

2. Please change the method name generateLinuxPrinterCommand() to generatePrinterCommand(). This will work for both Linux and UNIX and the shorter name is better for now. Put a comment in the javadoc to explain that the command is only suitable for Linux and UNIX.

3. Add class-level javadoc in PrinterStream.

4. I'd like you to format your code a little differently. Here is an example:

```
   public static int openPrinterStream(String name, StreamDaemon sd)
   {
      int streamId = -1;

      // TODO: do we need to make more appropriate check for linux/UNIX?
      boolean isWin = PlatformHelper.isUnderWindowsFamily();

      if (!isWin)
      {
         // Linux and UNIX

         if (processDaemon == null)
         {
            throw new IllegalStateException(
                     "ProcessDaemon is not initialized");
         }

         streamId = sd.store(new ProcessStream());

         processDaemon.launch(generatePrinterCommand(name), -1, streamId);
      }
      else
      {
         // windows
         streamId = sd.store(new NullStream());
      }

      return streamId;
   }
```

A summary of the changes:

- Shorter variable and method names means less code to read. It won't cause a problem as long as the names are still descriptive enough. It also makes it much easier to keep lines of code from breaking into multiple lines. Breaking code into multiple lines results in very messy code.
- Don't create local variables where there is no need.
- Add blank lines before and after blocks/groups of related code to make the code more readable.
- Put error checking up front to avoid doing unnecessary things when we are going to fail downstream anyway.

#68 - 09/23/2013 04:38 PM - Vadim Gindin

- File vig_upd20130923c.zip added

I made described corrections

#69 - 09/23/2013 05:13 PM - Greg Shah

OK, this is ready for testing.

#70 - 09/23/2013 05:38 PM - Greg Shah

Oops. I spoke too soon. Your P2J build is not clean:

antlr_progress:
[antlr] ANTLR Parser Generator Version 2.7.4 1989-2004 jGuru.com
[antlr] /home/vig/testing/majic/p2j/src/com/goldencode/p2j/uast/progress.g:16268:7: warning:nondeterminism between alts 1 and 2 of block upon
[antlr] /home/vig/testing/majic/p2j/src/com/goldencode/p2j/uast/progress.g:16268:7: k==1:STRING
[antlr] /home/vig/testing/majic/p2j/src/com/goldencode/p2j/uast/progress.g:16268:7:
k==2:DOT,KW_MAP,KW_NO_MAP,KW_APPEND,KW_BINARY,KW_COLLATE,KW_CONVERT,KW_ECHO,KW_KEEP_MSG,KW_LANDSCAP,KW_LOB_DIR,KW_NO_CVT,KW_NO_ECHO,KW_NUM_COPY,KW_PAGE_SZ,KW_PAGED,KW_PORTRAIT,KW_UNBUF
[antlr] /home/vig/testing/majic/p2j/src/com/goldencode/p2j/uast/progress.g:16268:7:
k==3:EOF,DOT,FILEROOT,DB_SYMBOL,ANY_LOCK,FILENAME,BACKSLASH,LIBRARY_REF,BEGIN_RECORDTYPES,TABLE,BUFFER,TEMP_TABLE,WORK_TABLE,END_RECORDTYPES,BEGIN_BUFFERTYPES,NO_REFERENCE,WEAK_REFERENCE,STRONG_REFERENCE,FREE_REFERENCE,END_BUFFERTYPES,EXPANDED_SCOPE,BUFFER_SCOPE,BEGIN_SCHEMA,ABBREVIATED,AREA,CYCLE_ON_LIMIT,DESCRIPTION,DUMP_NAME,FROZEN,HIDDEN,INCREMENT,MANDATORY,ORDER,POSITION,SQL_WIDTH,TRAILER,VALEXP,VALMSG,BEGIN_SCHEMAKW,KW_ABBV,KW_AREA,KW_BIGINT,KW_CLOB_CP,KW_CLOB_COL,KW_CLOB_TYP,KW_CRC,KW_CYCLE,KW_DESCR,KW_DMP_NAME,KW_FLD_TRG,KW_FIXCHAR,KW_FROZEN,KW_IDX_FLD,KW_INACTIVE,KW_INCR,KW_LOB_AREA,KW_LOB_BYTE,KW_LOB_SIZE,KW_NO_OVRRD,KW_NOT_CS,KW_NULLALWD,KW_PSC,KW_SEQUENCE,KW_SQL_WID,KW_TAB_TRG,KW_TIMESTMP,KW_VALEXP,KW_VALMSG,END_SCHEMAKW,BEGIN_METATYPES,FIELD_BIGINT,FIELD_BYTE,FIELD_DOUBLE,FIELD_FIXCHAR,FIELD_FLOAT,FIELD_SHORT,FIELD_TIMESTAMP,FIELD_TIME,END_METATYPES,END_SCHEMA,DATABASE,INDEX,INDEX_FIELD,BEGIN_INDEX_MATCH_TYPES,NO_MATCH,SELF_REFERENCE,EQUALITY_MATCH,BEGINS_MATCH,RANGE_MATCH,WORD_MATCH,SORT_MATCH,END_INDEX_MATCH_TYPES,SEQUENCE,PROPERTIES,MANY_TO_ONE,ONE_TO_MANY,ONE_TO_ONE,KEY_FIELD,RAW_STRING,QUERY,QUERY_SUBST,CLIENT_WHERE,CLIENT_WHERE_SUBST,EMBEDDED_SQL,COLUMN_LIST,BLOCK,PROCEDURE,CLASS_DEF,INTERFACE_DEF,METHOD_DEF,CONSTRUCTOR,DESTRUCTOR,CLASS_NAME,OBJECT_INVOCATION,EVENT_SIGNATURE,DOTNET_DELEGATE,INT_PROC,WEB_PROC,FUNCTION,INNER_BLOCK,TRIGGER_BLOCK,EDITING_BLOCK,FRAME_SCOPE,FRAME_ALLOC,COPY_TO_SB,COPY_FROM_SB,CUSTOMER_SPECIFIC,VALIDATION,TRANSACTION,SUB_TRANSACTION,NO_TRANSACTION,STATEMENT,LABEL_DEF,LABEL,EXPRESSION,EVENT,DB_EVENT,CLASS_EVENT,EVENT_LIST,KEY_FUNCTION,WIDGET_LIST,WIDGET_POOL,ASSIGNMENT,ASSIGN,ARGUMENTS,PARAMETER,COMMAND_TOKENS,COMMAND_TEXT,CONNECT_OPTIONS,CONNECT_TEXT,IO_OPTIONS,PUT_FIELD,IMPORT_FIELD,EXPORT_FIELD,AGGREGATE,FRAME_PHRASE,FORMAT_PHRASE,TRIGGER_PHRASE,RECORD_PHRASE,COLOR_PHRASE,CONTENT_ARRAY,FRAME_ELEMENT,WHEN_LIST,EMBEDDED_ASSIGNMENT,FORM_ITEM,COLUMN_REF,RUNTIME_FRAME_OPTIONS,RUNTIME_BROWSE_OPTIONS,PARENT_CHILD_RELATION,DATA_SET,DATA_SOURCE,DATA_RELATION,LANGUAGES,DYNAMIC,MINUS,PLUS,UN_MINUS,UN_PLUS,BOOL_TRUE,BOOL_FALSE,DEC_LITERAL,DATE_LITERAL,DATETIME_LITERAL,DATETIME_TZ_LITERAL,...

This is a sign that there is a conflict in progress.g. I have reviewed the code and I think it should be fine. So we must disable the warning like this (it is not acceptable to leave this kind of warning in our builds):

```
printer_clause
   :
      KW_PRINTER^
      (
         options { generateAmbigWarnings = false; }  <----------------- this line is added
         :                                           <----------------- this one too
           STRING
         |
           // avoid collisions with the following options
           {
              LA(1) != KW_BINARY   && LA(1) != KW_ECHO     &&
              LA(1) != KW_NO_ECHO  && LA(1) != KW_UNBUF    &&
              LA(1) != KW_MAP      && LA(1) != KW_NO_MAP   &&
              LA(1) != KW_NO_CVT   && LA(1) != KW_CONVERT  &&
              LA(1) != KW_NUM_COPY && LA(1) != KW_PAGE_SZ  &&
              LA(1) != KW_COLLATE  && LA(1) != KW_LANDSCAP &&
              LA(1) != KW_PORTRAIT && LA(1) != KW_APPEND   &&
              LA(1) != KW_PAGED    && LA(1) != KW_KEEP_MSG
           }?
           filename
         | // the empty alternative
      )
   ;
```

In addition, please apply the same cleanup items I mentioned in note 67 above to the PrinterStream.generatePrinterCommand(). The names are too long, there needs to be more blank space...

Finally, please add javadoc for the 2nd parameter of openPrinterStream().

#71 - 09/24/2013 08:58 AM - Vadim Gindin

- File vig_upd20130924a.zip added

I corrected progress.g and PrinterStream

#72 - 09/24/2013 11:08 AM - Greg Shah

OK, this is fine.

> In addition, please apply the same cleanup items I mentioned in note 67 above to the PrinterStream.generatePrinterCommand(). The names are too long, there needs to be more blank space...

By the way, you made more changes to openPrinterStream(), but didn't do any of the cleanup for generatePrinterCommand(). Generally, the changes to openPrinterStream() were not necessary. Removing the vowels from streamDaemon is not enough to really make it an improvement. Actually, now it may be too cryptic. Anyway, it is fine.

In the future, please follow the guidelines for all code that you write for us. The names in generatePrinterCommand() are a good example of code that is harder to read because of how long they are. It changes the formatting of the code around it (making it bigger and more bunched up than necessary, sometimes forcing it to cross lines). And it is more to read so it slows down the reader. The key is finding a balance between short names and something that is still descriptive.

It is OK to use very short names for local vars that are not being used over a large amount of code. "sd" instead of "streamDaemon" is OK in a short method where there are few vars and little code. It is not hard to remember that sd stands for streamDaemon in such a case.

I know we are very picky about the code. But we believe that getting it right is very important over the long term, to make the code as easy to read and maintain as possible. We have a very specific set of criteria that we have decided upon. Often this is different from common industry practice (e.g. curly braces on separate lines) or from personal preferences. I appreciate your patience and your commitment to writing code that fits this standard. Thanks!

#73 - 09/24/2013 04:23 PM - Vadim Gindin

- File vig_upd20130924b.zip added

Sure, I understand an importance of such rules. I just wanted to make a final step to remember goals of that rules. I've made a final update :)

Constantin have finished db creation and I'll be able to continue regression testing tomorrow.

#74 - 09/24/2013 04:37 PM - Greg Shah

It looks good. Thanks!

#75 - 10/10/2013 09:54 AM - Vadim Gindin

I've checked in the last update (vig_upd20130924b.zip). Revision number is 10396.

#76 - 10/10/2013 10:15 AM - Greg Shah

- Status changed from WIP to Closed

The update did pass regression testing.

#77 - 10/10/2013 10:49 AM - Vadim Gindin

I've committed the missed file PrinterStream. New rev is 10397.

#78 - 11/16/2016 12:07 PM - Greg Shah

- Target version changed from Milestone 11 to Cleanup and Stablization for Server Features

Files

cs_upd20130219a.zip	43.6 KB	02/19/2013	Costin Savin
cs_upd20130220a.zip	43.7 KB	02/20/2013	Costin Savin
cs_upd20130222b.zip	43.7 KB	02/22/2013	Costin Savin
output_to_printer_landscape_paged_150.pdf	6.95 KB	09/06/2013	Vadim Gindin
output_to_printer_landscape_paged_60.pdf	16.9 KB	09/06/2013	Vadim Gindin
output_to_printer_simple_row.p	209 Bytes	09/06/2013	Vadim Gindin
output_to_printer_simple_row.pdf	2.97 KB	09/06/2013	Vadim Gindin
vig_20130918a.zip	293 KB	09/17/2013	Vadim Gindin
vig_upd20130918b.zip	293 KB	09/20/2013	Vadim Gindin
vig_upd20130923a.zip	300 KB	09/23/2013	Vadim Gindin
vig_upd20130923b.zip	294 KB	09/23/2013	Vadim Gindin
vig_upd20130923c.zip	294 KB	09/23/2013	Vadim Gindin
vig_upd20130924a.zip	295 KB	09/24/2013	Vadim Gindin
vig_upd20130924b.zip	295 KB	09/24/2013	Vadim Gindin