User Interface - Feature #1787

implement batch mode support

10/30/2012 09:41 AM - Greg Shah

Status:	Closed	**Start date:**	06/10/2013
Priority:	Normal	**Due date:**	06/21/2013
Assignee:	Eugenie Lyzenko	**% Done:**	100%
Category:		**Estimated time:**	120.00 hours
Target version:	Runtime Support for Server Features		
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to Runtime Infrastructure - Feature #2124: improve/simplify the start...	**Closed**	**05/27/2013**	**06/07/2013**

History

#1 - 10/31/2012 02:54 PM - Greg Shah

- Target version set to Milestone 7

#2 - 04/16/2013 10:36 AM - Greg Shah

- Assignee set to Greg Shah

- Start date set to 06/10/2013

- Estimated time changed from 80.00 to 120.00

- Due date set to 06/21/2013

The purpose of this task is to duplicate those features of the 4GL that are enabled when the 4GL client is started with the -b parameter. Batch mode is also automatically enabled for appserver agents and webspeed agents. In all 3 cases, SESSION:BATCH-MODE will return TRUE.

The first step in this task is to determine the extent of this feature. The idea is to test each interactive 4GL feature in the following modes to determine the differences:

1. interactive, non-batch
2. redirected terminal, non-batch
3. no redirected terminal, batch mode
4. redirected terminal, batch mode

After that list of specific features is known, a design and implementation plan will be made. I expect 40 hours for the research and for now I will plan 80 more hours for the implementation.

#3 - 10/10/2013 11:16 AM - Greg Shah

- Assignee changed from Greg Shah to Eugenie Lyzenko

#4 - 10/10/2013 02:48 PM - Eugenie Lyzenko

> The purpose of this task is to duplicate those features of the 4GL that are enabled when the 4GL client is started with the -b parameter.
> ...
>
> The first step in this task is to determine the extent of this feature. The idea is to test each interactive 4GL feature

interactive 4GL feature - You mean for example the data field that updated by the user interaction in usual processing or imported from input files in redirected case? Or something else other?

#5 - 10/10/2013 03:43 PM - Greg Shah

> interactive 4GL feature - You mean for example the data field that updated by the user interaction in usual processing

I mean all 4GL features which block for user input. Examples:

UPDATE/SET/PROMPT-FOR (without input redirection)
CHOOSE stmt
INSERT stmt
ENABLE + WAIT-FOR stmts
SENSITIVE attribute
EDITING block with embedded READKEY + APPLY
READKEY statement (without input redirection)
PAUSE statement (this is an explicit pause in the 4GL)
implicit pause (e.g. MESSAGE statement that occurs after the 2 message lines are already full OR a DISPLAY of a DOWN frame that needs to show a row that would cause the output to scroll)
MESSAGE VIEW-AS ALERT-BOX (requires the user to select a button)
MESSAGE UPDATE/SET (this is a special kind of editing mode for a single var)
OS-COMMAND/UNIX... (when it is run with a program that requires user input)

> imported from input files in redirected case?

Input redirection cases are not considered interactive since they can execute with no real user input.

#6 - 10/20/2013 01:11 PM - Greg Shah

Please provide an update on your findings so far. I would also like to know how much more time you expect to need to finish the testcases/investigations.

#7 - 10/20/2013 02:28 PM - Eugenie Lyzenko

Basically I moved not far as expected taking into account additional time I had to spend to finish the CAN-QUERY/CAN-SET task.

But some finding already exist. I've started the testcases in batch mode, no redirection because this is most unknowing mode for this time.

All statements that require user interaction display the error message:

- Attempt to read with no current source of input. (513)
- Attempt to write with no current output destination.(516)
  depending on either input requested or trying to output by DISPLAY statement.

The only exception from this rule currently found id the PAUSE statement. In this case there is no error message and the program silently finishes.

The command was used to execute batch mode is: prowin32 -p testfile.p -b
The only way to suppress the error message is to redirect output to NULL device:
prowin32 -p testfile.p -b > null.

Continue investigation. Sorry for the additional delays, I'm expecting to finish testing in a next 2 days.

#8 - 10/20/2013 02:52 PM - Constantin Asofiei

> The only exception from this rule currently found id the PAUSE statement. In this case there is no error message and the program silently finishes.

I can't replicate this. By "the program silently finishes" do you mean that no code after the PAUSE is executed? Note that if I run the following program using "pro -p test.p -b" on lindev01:

```
pause 5.

message "bla".
```

5 seconds pass until the ** Attempt to write with no current output destination. (165) message is shown and program terminates. I just want to make sure that we are both on the same page. You might already be considering this, but please test the following too (beside what Greg mentioned in note 5):

- PUT SCREEN statement (I think this is a no-op in batch mode and no message is shown)
- what happens with error messages generated by certain statements/functions/methods/etc, when the NO-ERROR clause is **not** in effect.
- what happens when the stream uses the terminal for input/output? i.e. output stream s to terminal and input stream s from terminal.
- what happens when the unnamed stream is redirected back to the terminal? i.e. output to terminal and input from terminal.

#9 - 10/20/2013 03:12 PM - Eugenie Lyzenko

> By "the program silently finishes" do you mean that no code after the PAUSE is executed?

No I mean the program executed from start to end. The same situation is for
pause 10.
statement.

This code was executed on 4GL under Windows.

#10 - 10/20/2013 03:21 PM - Eugenie Lyzenko

The linvdev01 shows the same behavior as 4GL on Winodws:

```
pause. and
pause 10.
```

executed normally. But the command to start the batch program on Linux is a bit different than you use:

```
"bpro -p test.p -b"
```

#11 - 10/22/2013 07:26 PM - Eugenie Lyzenko

Constantin, what do you mean here:

> what happens with error messages generated by certain statements/functions/methods/etc, when the NO-ERROR clause is not in effect.

This means output/input with disabling error messages?

#12 - 10/23/2013 04:19 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

Constantin, what do you mean here:

> what happens with error messages generated by certain statements/functions/methods/etc, when the NO-ERROR clause is not in effect.

This means output/input with disabling error messages?

When the NO-ERROR clause is not used, an error message is displayed at the terminal when i.e. a FIND FIRST does not retrieve any records. In batch mode, we need to make sure the terminal will no be used.

#13 - 10/23/2013 07:21 PM - Eugenie Lyzenko

The finding results:

1. In general all statements that require users input or display output generate the error in batch mode without redirection:

```
** Attempt to read with no current source of input. (513)
** Attempt to write with no current output destination.(516)
```

This also includes: MESSAGE ALERT-BOX, MESSAGE UPDATE/SET, INSERT, EDITING block and CHOOSE.

2. However there are exceptions:

```
PAUSE n.
PUT SCREEN.
OS-COMMAND.
```

In these cases the program executes normally in batch mode. 3. Redirection with assigning stream target to TERMINAL like

```
output stream s to terminal
input stream s from terminal
redirecting unnamed stream to terminal
```

causes the error message in batch mode:

- There is no terminal(120)

4. The NO-ERROR modifier does not affect the batch mode behavior. We have the same error messages as for statement without NO-ERROR.

5. SENSITIVE attribute does not change the error messages behavior.

6. Redirected case, batch mode eliminate almost all errors except:
ENABLE statement generates the error:

- Cannot execute ENABLE statement when the output is not the screen (4018)

#14 - 10/24/2013 10:03 AM - Greg Shah

What happens in these cases:

1. The program's output is redirected to a file from the Progress command line:

```
_progress -p myprog.p -b > somefile.txt 2>&1
```

2. The program's input is redirected from a file or some other program from the Progress command line:

```
someprog | _progress -p myprog.p -b
```

3. The program's input and output is redirected from/to a file or some other program from the Progress command line:

```
someprog | _progress -p myprog.p -b > somefile.txt 2>&1
```

#15 - 10/24/2013 07:27 PM - Eugenie Lyzenko

```
_progress -p myprog.p -b > somefile.txt 2>&1
```

No error messages on the screen, only informations from 4GI interpreter itself like this:
Batch processing will be performed using:
OpenEdge Release 10.2B as of Mon Dec 14 17:00:18 EST 2009

However two errors are writing to the redirected file:
- Cannot execute ENABLE statement when the output is not the screen (4018)
- There is no terminal(120) - for statements that explicitly uses the terminal as i/o device

```
someprog | _progress -p myprog.p -b
```

SET does not generate the input errors as long as other statements requires only input.
The statements that use display as output device generates the errors.
Moreover the following errors are also happening
- Cannot execute ENABLE statement when the output is not the screen (4018)
- There is no terminal(120) - for statements that explicitly uses the terminal as i/o device

```
someprog | _progress -p myprog.p -b > somefile.txt 2>&1
```

We have the same status as for redirected case, batch mode:

- Cannot execute ENABLE statement when the output is not the screen (4018)
- There is no terminal(120) - for statements that explicitly uses the terminal as i/o device

#16 - 10/28/2013 04:50 PM - Greg Shah

I assume you are done with your testing. Based on your results, what do you propose as changes to implement this behavior?

Please note that we already support a batch mode flag in the directory.xml (in the runtime section). We will also want to allow a per-client override in the bootstrap configuration.

Once the client "knows" that it is in batch mode, what are the changes that need to be made to duplicate the behavior of batch mode?

#17 - 10/28/2013 05:28 PM - Eugenie Lyzenko

> I assume you are done with your testing. Based on your results, what do you propose as changes to implement this behavior?

Yes, I'm thinking on suggested approaches.

> Please note that we already support a batch mode flag in the directory.xml (in the runtime section). We will also want to allow a per-client override in the bootstrap configuration.

And here I need some clarification for complete understanding for what is the difference between existing approach and what we want to have now(from the client starting point of view)? In a new approach with the only command client.sh we need the ability to start either interactive mode or batch mode? We need to have the batch mode ability without specially created run/batch/* directory item? Or we need to modify the existed implementation to be exact as the current 4GL batch mode?

#18 - 10/28/2013 07:12 PM - Greg Shah

> In a new approach with the only command client.sh we need the ability to start either interactive mode or batch mode?

Please see EnvironmentOps.isBatchMode() and EnvironmentOps.setBatchMode(). These are backed by the directory today.

I would like to add the ability to set a value in the bootstrap configuration (either in the bootstrap config XML file OR via the command line) to force batch mode for the client. Is is IN ADDITION TO the setting that may already be in the directory for a given account.

For example, if you have a BootstrapConfig instance named bc, you could determine if it was in batch mode using this:

boolean batch = bc.getBoolean("client", "mode", "batch", false);

Once we have this BootstrapConfig support (IN ADDITION TO the existing directory support), then the client code (ThinClient?) would need to be configured to honor batch mode. This could be done in TC.initializePrep() using the BootstrapConfig BUT that is not a good place to override batch mode by reading from the Directory instance.

Anyway, if we add support for reading from the BootstrapConfig instance, then the client.sh can already override the batch mode value from the command line by adding this to the end of the command line: client:mode:batch=true.

Enabling batch mode through configuration is the smaller part of this task.

> We need to have the batch mode ability without specially created run/batch/* directory item?

Yes.

> Or we need to modify the existed implementation to be exact as the current 4GL batch mode?

Yes, this also needs to be done. And this is the bigger part of the task. When batch mode is enabled, the client must operate exactly as in the 4GL batch mode. So every UI feature (and error processing) should be batch mode aware as needed.

#19 - 10/30/2013 05:05 PM - Eugenie Lyzenko

So my suggestions at this time:

1. For bootstrap config. We can declare using the batch mode either in client.xml file or command line with client starting script client.sh. All these two options will be equivalent but in the case we have both of them the option from the starting script will have highest priority. At the time the client configuration is starting - ClientCore.java class all differences will be lost we will just "know" we are in batch mode on client side if my understanding of the task requirements are correct.

2. The main part of modification is the full 4GL simulation of the batch mode behavior. Agree, the ThinClient itself is a not a good place to implement this. Generally the main difference for batch mode is we really missing two objects - output and input device. So instead of tracking all statements/widgets/errors for requirements to generate possible i/o error, I would suggest to create special screen driver for batch mode - like ConsoleDriver, something like DummyDriver and DummyPrimitives. This way we concentrate all i/o modifications in single place providing i/o calls stubs with respective errors generations having ThinClient code and Widget's related code untouched. Or we can even separate input from output having dummy input and real output and vice versa. So when the ClientCore honors the batch mode the OutputManager initializing(in ClientCore.start() and ThinClient.initializePrep()) special screen driver starts working instead of ConsoleDriver or SwingDriver. This special driver will be initialized completely in OutputManager class. And we do not need any native calls implementation for this batch mode driver.

#20 - 10/30/2013 05:16 PM - Greg Shah

> We can declare using the batch mode either in client.xml file or command line with client starting script client.sh. All these two options will be equivalent but in the case we have both of them the option from the starting script will have highest priority.

To be clear, there is nothing you need to do to add this functionality. The ability to pass this value (from the .xml file or on the command line) is already there.

> At the time the client configuration is starting - ClientCore.java class all differences will be lost we will just "know" we are in batch mode on client side if my understanding of the task requirements are correct.

All you need to do is read the config value from the bootstrap config in ClientCore.

> So instead of tracking all statements/widgets/errors for requirements to generate possible i/o error, I would suggest to create special screen driver for batch mode - like ConsoleDriver, something like DummyDriver and DummyPrimitives. This way we concentrate all i/o modifications in single place providing i/o calls stubs with respective errors generations having ThinClient code and Widget's related code untouched.

I like the idea in general, but I suspect that there is behavior that cannot be hidden inside the driver. For example, the editing statements generate errors when not operating on redirected input but the driver has no knowledge of whether we are editing or not.

Also, to the extent that behavior changes based on redirected input and/or output, we may want to setup the redirection via the unnamed streams if that can properly duplicate the behavior. In other words, I hope we can support some of the behavior using features we already have.

This last point brings up another question: what happens if you run code in batch mode that redirects the unnamed streams to some other file destination? Will the 4GL honor this code? I suspect (and hope) so.

#21 - 10/30/2013 05:54 PM - Eugenie Lyzenko

> This last point brings up another question: what happens if you run code in batch mode that redirects the unnamed streams to some other file

destination? Will the 4GL honor this code? I suspect (and hope) so.

When unnamed stream is redirected, the only error case is using ENABLE statements:

```
Cannot execute ENABLE statement when the output is not the screen (4018)
```

#22 - 10/30/2013 06:04 PM - Greg Shah

I am wondering what happens if you have the input and/or output of the process redirected from the command line and then inside the 4GL code, that 4GL code then changes the redirection by redirecting the unnamed stream(s) to some other place.

#23 - 10/30/2013 06:21 PM - Eugenie Lyzenko

> I am wondering what happens if you have the input and/or output of the process redirected from the command line and then inside the 4GL code, that 4GL code then changes the redirection by redirecting the unnamed stream(s) to some other place.

Are you talking about the redirected code like program with:

```
INPUT FROM "somefile.txt".
OUTPUT TO "anotherfile.txt".
...
INPUT CLOSE.
OUTPUT CLOSE.
```

running in batch mode?

or procedure that performs redirection on command line level:

```
someprog | _progress -p myprog.p -b > somefile.txt 2>&1
```

These two cases can be not the same from the 4GL point of view. And I can suggest they probably can be different for Linux and Windows due to different redirection system implementation.

#24 - 10/30/2013 06:26 PM - Greg Shah

I am talking about if the 4GL code is using both techniques at the same time. So a program started like this:

```
someprog | _progress -p myprog.p -b > somefile.txt 2>&1
```

Has 4GL code called from myprog.p that does this:

```
INPUT FROM "somefile.txt".
OUTPUT TO "anotherfile.txt".
...
INPUT CLOSE.
OUTPUT CLOSE.
```

It is the combination that I worry about.

> These two cases can be not the same from the 4GL point of view.

I suspect they are actually the same thing in the 4GL.

> And I can suggest they probably can be different for Linux and Windows due to different redirection system implementation.

Possibly. Please test to find out.

#25 - 10/31/2013 02:31 PM - Eugenie Lyzenko

OK. The findings are:

1. 4GL behaves the same way in Linux and Windows. The difference is not related to the i/o or pipe system redirection. There is one difference: The batch mode 4GL in Linux gives the following output:

```
Batch processing will be performed using:
OpenEdge Release 10.2B as of Mon Dec 14 17:00:18 EST 2009
...
```

In case of redirecting(... > somefile.txt 2>&1) this is the content of the somefile.txt.

2. The application result is:
- because inside application unnamed streams are the targets of the i/o - all input output are redirected to the files defined inside application.
- external redirection(... > somefile.txt 2>&1) here means all output of the 4GL program in batch mode(no output if we redirected i/o internally inside the program) will be put into somefile.txt. As the result: Windows has empty file somefile.txt, Linux put the OpenEdge Release version message to this file.

#26 - 10/31/2013 02:46 PM - Greg Shah

Good, this helps to clarify things. Notice how only certain interactive cases cause errors and the rest of the I/O subsystems continue working unchanged. I assume that UI things like DISPLAY still work with redirected unnamed streams output. Is that correct?

#27 - 10/31/2013 03:03 PM - Eugenie Lyzenko

> I assume that UI things like DISPLAY still work with redirected unnamed streams output. Is that correct?

Yes, these statements in batch mode output to file as redirected target without error messages.

#28 - 11/05/2013 04:52 PM - Eugenie Lyzenko

- File evl_upd20131105a.zip added

This is the current status update for what I'm working on for you to keep track.

The update includes prototype for batch mode implementation. The idea is to use the current console based driver at maximum where the functionality is the same. The newly created batch driver, helper and primitives are derived from console counterparts. When the batch mode is not in action the driver calls console based one. The console driver is used as the base because the batch mode is actually started from console. The code is very rough just to present the idea for now.

For now I've found the P2J calls the following API in console mode:
updateScreen()
cursorAt()
sync()
These calls should be different in batch mode, and
convertToBuffer()
encode()
character()
attribute()
color()
These calls can be same in console interactive mode and batch mode.

The active issues to work on:
1. The batch mode should be started to work after all usual client's preliminary steps are done. I have included special flag into BatchPrimitives to be

able to turn batch functionality on. The current implementation is certainly not good.
2. There is a conflict between ErrorManager messages and the messages we need to display in batch mode. Also if we use ErrorManager.showErrorAndAbend() the client is not terminating as it happens in 4GL. The client is restarting form beginning instead.
3. There should be special processing for statements like PAUSE.

Continue working.

#29 - 11/10/2013 05:54 PM - Eugenie Lyzenko

- File evl_upd20131109a.zip added

The further investigations and debugging shows the following requirements with the current approach:

1. The real batch mode in 4GL does not touch the current screen content while our P2J library clears the entire screen on initializing. So we need to bypass the console init code the preserve the compatibility with 4GL behavior. This is not a big issue because in batch mode it is supposed the console is not used for input/output operations in 4GL. But we still need to use some functions from console driver - getting ScreenBitmap() object for possible redirecting case usage. I'm making the list of tools we need to use from current console driver in a new batch driver.
2. The usage of the ErrorManager output functions conflicts with missing of the output device. So we need to use another tool to output such messages. However we can use ErrorManager to build the message text.
3. The generation of the QuitConditionException now used to stop the application because abnormal end causes the application to be restarted instead of quit.

The next step is to find out the possibility to handle input for UPDATE, PROMPT-FOR and other statements that require the user input and to handle the redirected i/o cases.

#30 - 11/11/2013 12:40 PM - Greg Shah

Code Review 1109a

1. I like the approach. Keep going with this.

2. What is the purpose of the BatchPrimitives.batchModeOn flag? It doesn't really seem to mean that we are in batch mode. Instead, it seems to be used to allow only 1 error to be reported for the session. Is that right? If so, then we need a different name that could describe this role better.

3. Is it sure that in batch mode the terminal size is always reported as 80x24?

4. Make sure that the SESSION:BATCH-MODE attribute will report the correct value (true) when we use the BatchDriver.

5. In regard to the ErrorManager case, please make a proposal for discussion. For a long time, I have wanted to abstract the terminal message processing from the ErrorManager implementation. In other words, I really dislike that we have hard coded UI references in the ErrorManager. I would prefer an approach where we create a simple Java interface that provides the output services needed by ErrorManager. Then the specific client implementation would "register" an object that implements that interface, and the ErrorManager would not have direct references to the UI. This might give you the tools you need to intercept/handle the output properly.

6. I need more information about your plans for the QuitConditionException. The abnormal end causes the application to be restarted instead of quit behavior is implemented based on the stopDisp variable in StandardServer. Please review that logic to understand what is going on.

#31 - 11/11/2013 01:45 PM - Eugenie Lyzenko

> 2. What is the purpose of the BatchPrimitives.batchModeOn flag? It doesn't really seem to mean that we are in batch mode. Instead, it seems to be used to allow only 1 error to be reported for the session. Is that right? If so, then we need a different name that could describe this role better.

Yes you are right, this flag does not mean we are in batch mode because we are certainly in batch mode. The purpose is to determine the moment when we have to display error in response to the i/o request. It should not always be true. For example on the application init stage when program calls methods from current batch driver or when the statements like PAUSE is executing we need to ignore the calls or pass the control to the console driver not displaying the error.

> 6. I need more information about your plans for the QuitConditionException. The abnormal end causes the application to be restarted instead of quit behavior is implemented based on the stopDisp variable in StandardServer. Please review that logic to understand what is going on.

OK. I'll review this. For now I use this exception because the 4GL really stops the application when facing with i/o statements in batch mode. I just reproduced the 4GL behavior here.

#32 - 11/11/2013 02:22 PM - Greg Shah

> For now I use this exception because the 4GL really stops the application when facing with i/o statements in batch mode. I just reproduced the 4GL behavior here.

Please test such a case with the following code:

```
do on stop undo leave:
   /* failing statement goes here */
end.

message "caught the stop condition".
```

If you don't see the message, then the failure doesn't raise a STOP condition. You can then change the stop to quit and re-test. The point is that you can determine which condition is raised.

If it is the STOP condition, then additional changes may be needed to match your results.

#33 - 11/12/2013 06:30 PM - Eugenie Lyzenko

The strange results have been gotten for the following testcase:

```
def var chVar as character.

do on stop|quit undo, leave:
   /* failing statement goes here */
   readkey. (Windows)
   update chVar. (Linux)
end.

output to "batch_test.log".
message "caught the stop condition".
output close.
```

The Windows and Linux show both the same results for using STOP or QUIT criteria. If the failing occurs the message "caught the stop condition". statement is not executed. Looks like the applications just stops exactly at the point where the first error is happening. The log file is not even created. No matter either STOP or QUIT was specified. If the error is ignored(as for "readkey" statement in Linux) - the message is going to the log file. And again no matter if STOP or QUIT was used.

#34 - 11/13/2013 10:14 AM - Greg Shah

Please try ERROR instead of STOP or QUIT.

#35 - 11/13/2013 03:08 PM - Eugenie Lyzenko

> Please try ERROR instead of STOP or QUIT.

The same result as for STOP and QUIT.

#36 - 11/13/2013 03:10 PM - Greg Shah

I guess it is worth trying ENDKEY too.

#37 - 11/14/2013 11:05 AM - Eugenie Lyzenko

The ENDKEY works the same way as ERROR, STOP, QUIT. I have tested even interception by CATCH block usage like:

```
...
   CATCH eAppError AS Progress.Lang.AppError:
      OUTPUT TO "batch_test.log".
      MESSAGE "Progress.Lang.AppError catched".
      OUTPUT CLOSE.
   END CATCH.

   CATCH eSysError AS Progress.Lang.SysError:
      OUTPUT TO "batch_test.log".
      MESSAGE "Progress.Lang.SysError catched".
      OUTPUT CLOSE.
   END CATCH.

   CATCH eError AS Progress.Lang.Error:
      OUTPUT TO "batch_test.log".
      MESSAGE "Progress.Lang.Error catched".
      OUTPUT CLOSE.
   END CATCH.
...
```

No result. The application just stops at the point when failure happening. This is confirmed by description on the page 52 in dverr.pdf OpenEdge 11.1 documentation. I think the application should return control to the calling procedure(or to OS when there is no nested calls).

#38 - 11/14/2013 12:16 PM - Greg Shah

No, what they are describing on page 52 is the ERROR condition. We already support that fully.

You have found something else: an uncatchable condition.

Questions:

1. If you call the failing program from another .p file, does the calling .p continue to run after the "failed" .p returns?
2. Is there any message on the screen or in any redirected log file when this occurs?

#39 - 11/14/2013 03:45 PM - Eugenie Lyzenko

OK. I have got the strange results:

1. Calling program does not continue after failed returns.
2. The answer no because the program stops at the failure point.

3. The 1 and 2 are the same for both Windows and Linux and for every cases: ENDKEY, ERROR, STOP and QUIT.

So this is really interesting. Is there any way to find out what is happening in 4GL internally? Looks like we need something like Progress 4GL "source level debugger".

#40 - 11/14/2013 05:25 PM - Greg Shah

> Looks like we need something like Progress 4GL "source level debugger".

Progress does have a debugger. I've never used it and I'm not sure how to do so (you'll have to read the docs).

1. Calling program does not continue after failed returns.

Try one last thing. In the calling program, put this code in:

```
do on error undo, leave
   on endkey undo, leave
   on stop undo, leave
   on quit undo, leave:

   run called_program.p.
end.

message "after call in caller".
```

#41 - 11/14/2013 06:02 PM - Eugenie Lyzenko

No changes for both Windows and Linux and for every cases: ENDKEY, ERROR, STOP and QUIT. The tests are:
Caller:

```
do on error undo, leave
   on endkey undo, leave
   on stop undo, leave
   on quit undo, leave:

   run batch_test16_1.p.
```

```
end.

OUTPUT TO "batch_test.log" APPEND.
MESSAGE "caught the stop condition".
OUTPUT CLOSE.
```

Calling program:

```
DEF VAR chVar AS CHARACTER.

DO ON ENDKEY|ERROR|STOP|QUIT UNDO, LEAVE:

   UPDATE chVar.
   readkey.

   CATCH eAppError AS Progress.Lang.AppError:
      OUTPUT TO "batch_test.log".
      MESSAGE "Progress.Lang.AppError catched".
      OUTPUT CLOSE.
   END CATCH.

   CATCH eSysError AS Progress.Lang.SysError:
      OUTPUT TO "batch_test.log".
      MESSAGE "Progress.Lang.SysError catched".
      OUTPUT CLOSE.
   END CATCH.

   CATCH eError AS Progress.Lang.Error:
      OUTPUT TO "batch_test.log".
      MESSAGE "Progress.Lang.Error catched".
      OUTPUT CLOSE.
   END CATCH.

END.
```

#42 - 11/15/2013 09:48 AM - Greg Shah

OK, so we have some kind of uncatchable abend.

Please post the list of the exact cases (e.g. READKEY on LINUX) which cause this in batch mode.

To implement this special case, please create a new class:

```
public class UnstoppableExitException
extends RuntimeException
```

You will need to add logic to StandardServer.standardEntry() to catch this new exception and set the mainResult to false. Do NOT rethrow the exception, otherwise we will treat this as an unexpected abend (and log the problem) when this is really a special kind of immediate exit from Progress.

#43 - 11/15/2013 04:29 PM - Eugenie Lyzenko

> You will need to add logic to StandardServer.standardEntry() to catch this new exception and set the mainResult to false. Do NOT rethrow the exception, otherwise we will treat this as an unexpected abend (and log the problem) when this is really a special kind of immediate exit from Progress.

OK. However we need to modify one more place in StandardServer class. The method invoke() should catch the UnstoppableExitException like this:

```
...
catch (Throwable th)
{
   Throwable chained = th;

   boolean   isError = false;
   boolean   isStop  = false;

   // look through all causes to determine if there was an
   // underlying error
   while (chained != null)
   {
...
      if (chained instanceof UnstoppableExitException)
      {
         throw new UnstoppableExitException(chained);
      }
...
      chained = chained.getCause();
}
...
```

Otherwise throwing UnstoppableExitException on the client side causes the StopConditionException on the server side and this will cause catching StopConditionException in StandardServer.standardEntry() and returns mainResult as true which in turn restarts the application instead of immediate halt the program.

#44 - 11/17/2013 04:36 PM - Eugenie Lyzenko

Both Linux and Windows generates error:

```
MESSAGE ALERT-BOX
MESSAGE UPDATE/SET
INSERT
EDITING block
CHOOSE
OUTPUT STREAM S TO TERMINAL
INPUT STREAM S FROM TERMINAL
UPDATE
```

Windows generates error, Linux - not

```
READKEY
SET
PROMPT-FOR
```

Linux generates error, Windows - not.

```
ENABLE
```

It is strange result but looks line Linux silently ignores the statements with input requests(READKEY, SET,...).

#45 - 11/17/2013 05:24 PM - Eugenie Lyzenko

5. In regard to the ErrorManager case, please make a proposal for discussion. For a long time, I have wanted to abstract the terminal message

> processing from the ErrorManager implementation. In other words, I really dislike that we have hard coded UI references in the ErrorManager. I would prefer an approach where we create a simple Java interface that provides the output services needed by ErrorManager. Then the specific client implementation would "register" an object that implements that interface, and the ErrorManager would not have direct references to the UI. This might give you the tools you need to intercept/handle the output properly.

I would suggest to add new interface defining the following methods.
displayError()
pause()
message()
messageBox()

Then ClientCore will create new object implementing these methods and initializes the ErrorManager with instance of this. Then ErrorManager will use this object to handle the calls. This is concept.

One note here. Windows uses standard Window's dialogue(Graphical window with OK button) to display the error even if the client starts in CHUI mode on Windows. So we have to identify the OS when creating the error messages handler. In Linux I guess we can use stdout or stderr channels.

#46 - 11/18/2013 11:19 AM - Greg Shah

I am generally OK with the proposal. I'll need to see the implementation to comment further.

> One note here. Windows uses standard Window's dialogue(Graphical window with OK button) to display the error even if the client starts in CHUI mode on Windows. So we have to identify the OS when creating the error messages handler. In Linux I guess we can use stdout or stderr channels.

Are you talking about batch mode only here? In non-batch mode, we want to reuse the ThinClient implementations. Even in batch mode, we want to use the ThinClient support as much as is possible. For example, we will already be implementing a messageBox() implementation for Windows.

I'll need to think about the point about Windows behaving differently. In our case, it may be more important to mimic the Windows behavior when the original application was running on Windows rather than based on where the P2J client is running. For example, it will be possible to run a 4GL "Windows" app as a converted Swing app in Linux.

#47 - 11/19/2013 08:29 PM - Eugenie Lyzenko

- File evl_upd20131119a.zip added

This update for review includes implementation proposal for interface implementing the error messages output for different client types. The concrete instance is creating in ErrorManager.initClient() method. The OutputManager includes the flag to indicate if we have override the batch mode flag(if it happens in client bootstrap configuration).

The new classes has still no javadoc comments because the changes are highly possible.

#48 - 11/23/2013 04:14 PM - Eugenie Lyzenko

The question. I'm working on PAUSE statement handling for batch mode and other statements like this that silently ignored in this mode. The idea is to deactivate error generation in a head of the ThinClient.pause() method and activate at the end of this statement. Otherwise we will have to know in BatchDriver what statement generates i/o methods in screen driver.

Fortunately in addition to PAUSE we have PUT SCREEN with similar effect I guess. So the list of the ignoring statements is small.

#49 - 11/24/2013 08:13 AM - Greg Shah

The idea makes sense.

#50 - 11/24/2013 08:47 AM - Greg Shah

Code Review 1119a

This is a move in the right direction, but there are some important changes to make:

1. The ErrorWriter interface must NOT have any references to com.goldencode.p2j.ui.* or any UI related sub-package. This means that the interface itself must change so that there is no external reference to the ClientExports. The implementation of the interface MUST HIDE the direct access to the "real client" or whatever output medium is appropriate.

2. The ErrorWriterInteractive should be in the UI package.

3. The ErrorWriterBatch can probably stay in util for now but it might also be moved to stay with ErrorWriterInteractive.

4. The "registration" of the proper ErrorWriter should be done by ClientCore or OutputManager, where we setup the primitives.

5. I assume that when 4GL code does SESSION:BATCH-MODE = true. or SESSION:BATCH-MODE = false. that the 4GL can dynamically shift into and out of batch mode? If so, then we need to implement this feature too. This will require the shift into/out of batch mode to be made modular/exposed as a method.

6. The ErrorManager class must NOT have any references to com.goldencode.p2j.ui.* or any UI related sub-package.

Good work. Keep going. Let's try to finish this in the next few days, OK?

#51 - 11/25/2013 09:26 AM - Eugenie Lyzenko

The question.

6. The ErrorManager class must NOT have any references to com.goldencode.p2j.ui.* or any UI related sub-package.

But ErrorManager already had the references to:

```
...
import com.goldencode.p2j.ui.*;
import com.goldencode.p2j.ui.chui.ThinClient;
...
```

This is the state before I started to work on batch mode. And the client member is used in ErrorManager class. Do we need to rework this too?

Let's try to finish this in the next few days, OK?

OK. I'll prepare the changes ASAP.

#52 - 11/25/2013 10:14 AM - Greg Shah

This is the state before I started to work on batch mode. And the client member is used in ErrorManager class.

Yes, I know. We are talking this opportunity to clean things up. They key here is that we want to reduce dependencies between the util package and the UI packages.

Do we need to rework this too?

Yes, the described approach should eliminate this dependency.

#53 - 11/25/2013 01:08 PM - Eugenie Lyzenko

> Yes, the described approach should eliminate this dependency.

OK. I have cleaned almost everything UI related from ErrorManager class. The only remaining reference is LogicalTerminal for messageBox()/message()/pause(). Correct me if I'm wrong but looks like this is the server related calls.

So do we need to have two separate error writer implementations, one for client side and one for server side? Or we can use the client's writer to display server errors because after server is up and running if there is no client - there will be no runtime errors to display.

What do you think? Of course the more clean way is to have two error writers for both client|server sides.

#54 - 11/25/2013 01:28 PM - Greg Shah

> So do we need to have two separate error writer implementations, one for client side and one for server side?

Yes, we need 2 versions. This is because we can generate errors on either the server or the client, but in both cases they need to be displayed at the client.

> Or we can use the client's writer to display server errors because after server is up and running if there is no client - there will be no runtime errors to display.

No.

#55 - 11/25/2013 02:48 PM - Eugenie Lyzenko

- File evl_upd20131125a.zip added

The update for your review. Includes:
1. ErrorWriter is cleaned from UI related classes.
2. ErrorWriter* implementations are all now in UI package. The new separate implementation is created for server side.

3. The client writer part "registration" is moved to OutputManager. I thought to do this in ClientCore first but this class has reference to ErrorManager from another package(java.util.logging.*) so to avoid confusion the error writer init is moved to OutputManager.
4. The server side error writer "registration" is performing in StandardServer class.
5. The ErrorManager class is now free from com.goldencode.p2j.ui.* or any UI related sub-package.

Continue working(not forgetting about Javadoc comments)...

#56 - 11/27/2013 05:19 PM - Eugenie Lyzenko

- File evl_upd20131127a.zip added

The new drop for review.
1. Completed Javadoc comments.
2. Added handling of PAUSE like processing when we need to temporary deactivate batch mode error generation. The ThinClient has been modified for this. My previous suggestion for approach to deactivate on PAUSE start and activate again on PAUSE finish was inconsistent because there are many calls i/o between PAUSE and new i/o statements than need the batch mode is still inactive for proper processing(discovered in java debug sessions). So we have to re-activate only on start of the new statement that need to be handled with active batch mode. That's why it is activated in ThinClient.promptFor() call.
3. The special key processing was introduced. In active batch mode we need to tell key reader to ignore key input channel - I've used Key.VK_F1 to go further to avoid endless cycle of key reading in TypeAhead class.

The drop still has EVL*** markers to easily find the changes I made.

Continue working.

#57 - 11/29/2013 09:04 AM - Eugenie Lyzenko

> 5. I assume that when 4GL code does SESSION:BATCH-MODE = true. or SESSION:BATCH-MODE = false. that the 4GL can dynamically shift into and out of batch mode? If so, then we need to implement this feature too. This will require the shift into/out of batch mode to be made modular/exposed as a method.

These statements are currently not supported on conversion level. The line

```
...
SESSION:BATCH-MODE = false.
or
SESSION:BATCH-MODE = true.
...
```

Both converted to:

```
...
SessionUtils.readOnlyError("BATCH-MODE");
...
```

This means the SESSION:BATCH-MODE attribute has read-only access, not changeable, correct me if I'm wrong. So do we really need the wright support for this attribute now?

#58 - 11/30/2013 08:18 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> This means the SESSION:BATCH-MODE attribute has read-only access, not changeable, correct me if I'm wrong. So do we really need the wright support for this attribute now?

The BATCH-MODE attribute is read-only; that's why the conversion rules will emit a readOnlyError call for it.

#59 - 12/02/2013 06:21 PM - Eugenie Lyzenko

- File evl_upd20131202a.zip added

This update adds OS-COMMAND support for batch mode. Continue testing for all problematic statements/OS

#60 - 12/05/2013 10:42 AM - Eugenie Lyzenko

The status update. Almost all statements are verified. The current work is to handle OUTPUT TO TERMINAL statement with There is no terminal(120) message. The problem is the moment the application should stop and the message to display is when the servers side trying to assign the output to terminal. So here we have to:
1. Check if the client is running in batch mode
2. Display the message on client's terminal
3. Stop the client.

All these action should be performed from the client side. I have the solution for everything except displaying the message itself. The server side ErrorWriter put the message into stdout.log if using System.out.println. But we can not use LogicalTerminal.message() because this means we need to use non-batch session(like for MESSAGE statement itself). So we need to perform System.out.println but on the client side and to introduce several new methods in LogicalTerminal to display the messages for this special case. In addition this approach will be used for the every case when ErrorManager should display the server-side messages using the client-side output.

#61 - 12/05/2013 06:03 PM - Eugenie Lyzenko

- File evl_upd20131205a.zip added

This update adds stream cases messages and stops for the clients. In addition to error writer implementation when server side error writer uses client's writer to display the fatal error messages. And yes, the stream related "There is no terminal (120)" message does not have "**" prefix.

Expecting to finish OS specific checking tomorrow.

#62 - 12/07/2013 05:31 PM - Eugenie Lyzenko

- File evl_upd20131206a.zip added

After testing the update for review. What new/changed:
1. The requirement to separate SET/PROMPT-FOR/UPDATE reaction for different OS cause moving the control for application stop on the server side(GenericFrame.java).
2. On Windows the client displays the errors using Java Swing message box.
3. On Linux the batch driver displays info message every time the application starts.
4. The "There is no terminal (120)" message really has the "**" prefix.

#63 - 12/08/2013 03:32 PM - Eugenie Lyzenko

- File evl_upd20131208a.zip added

The GenericFrame.java has been merged with the recent code base.

#64 - 12/10/2013 08:50 PM - Eugenie Lyzenko

- File evl_upd21031210a.zip added

This update includes some fixes and improvements to make it working under Windows. In this case we have to fully initialize the P2J.DLL subsystem instead of just loading library. For Linux we avoid this because initChui() call in driver clears the terminal screen and this is not happening in Linux 4GL batch. In Windows the batch mode does not use console to display the error so this is not critical and in Windows case we need the key transformations - so full CHUI initialization is required.

Also when we initialize stop the client from the server side(GenericFrame.java class) we need to deactivate the batch special processing from GenericFrame code to avoid message duplication in Windows.

#65 - 12/11/2013 10:05 AM - Greg Shah

Code Review 1210a

1. In StandardServer, the call to ErrorManager.initErrorWriterServer() should be placed in registerDefaultServices() where we do the rest of our 4GL-specific service initialization. We try to have the bootstrap() method be more generic (not specific to the 4GL), so that later we can refactor this more easily.

2. It is not clear from the code why the enabling/activation of batch mode is done in such a complicated manner. For example, why isn't the batch driver always started in batch mode? Instead we do this activation process using OutputManager.activateBatch() (which is a confusing name for the method). Calling that method with a true parameter in ClientCore suggests that we are always hard-coded to batch mode, but you have to read the activateBatch() code to find out that it really isn't doing that in all cases. I want to simplify this unless it is absolutely needed. If it is needed, we need better documentation explaining why we do this in multiple steps AND we need to rename activateBatch() to something that matches more accurately with the real functionality.

3. One of the ways we tell if we should be in batch mode is we check if the driver name is the batch driver name. I prefer if the driver interface just had a simple isBatchMode() method that the batch driver could just implement by returning true.

4. I don't see why there are both clientErrorOut and serverErrorOut members in ErrorManager. We only need one of these since we only ever have one or the other, right? If we need a way to tell if we are on the server or not, then find a better way to do that one thing. But there is no reason to have 2 instances of the ErrorWriter.

5. ErrorWriterInteractive is only ever used on the client side. It doesn't need any of the LogicalTerminal code, right (that stuff is already handled by the ErrorWriterServer class)?

6. We can't use a static reference to ThinClient in ErrorWriterInteractive. The reason is that there are times at which we "restart" the client and the ThinClient instance is replaced. Please switch to a single constructor for ErrorWriterInteractive that accepts an instance of ThinClient as a parameter. Then instead of a static member for client, it would be an instance member. In addition, make sure that when ErrorManager.resetClient() is called, that we create a new instance of ErrorWriterInteractive that refers to the new ThinClient instance.

7. You are adding some new dependencies to the ui package from the util package. We are trying to eliminate such dependencies, right? So, it seems to me that the processing in classes like StreamWrapper that call LogicalTerminal, should instead call something in ErrorManager and the ErrorWriter interface should be extended to provide the necessary support.

8. What is left to do in this task? Please post a list here.

#66 - 12/11/2013 08:48 PM - Eugenie Lyzenko

1. OK. Understood.

2. Yes, agree. this is not so simple as we want it to be. But actually the UI code is complicated itself(for console driver itself). We have multiple UI calls for single 4GL statement. So handling the UI requests we can not skip/substitute only single call letting other to be handled as usual because the UI code methods are very coupled one with other. So the approach is to select the code blocks we need to ignore and substitute with batch handling code whole block. That'a why I use the term "active" for batch driver. This means the special processing for i/o requests. When inactive we need to pass the control to the underlying console driver layer to provide code consistency and handle redirected cases(when the batch driver works as usual console driver).

3. OK, agree. I'll add this method to the driver interface.

4. OK. I'll find the better solution.

5. OK.

6. OK.

7.

> You are adding some new dependencies to the ui package from the util package. We are trying to eliminate such dependencies, right?

Correct. But the StreamWrapper and UnnamedStreams are the places where we have to intercept the stream requests in batch mode to stop the client. We can not do this later on the client side for example. Correct me if I'm wrong but for the client side the moment of stream assigning is unknown and the client working with already opened streams.

> So, it seems to me that the processing in classes like StreamWrapper that call LogicalTerminal, should instead call something in ErrorManager and the ErrorWriter interface should be extended to provide the necessary support.

In this case the ErrorManager class will contain the methods to identify the batch mode and to stop the client itself. Is it OK to have these client specific methods in ErrorManager/ErrorWriter?

8. In addition to make the changes for the above point I thought about the following:
- Testing for redirected cases in batch mode
- Need to think what to do with error display in Windows. Currently I've used Swing dialogue to implement this. But in Windows 4GL the dialogue is native. So probably we need to change P2J.DLL adding respective JNI. What do you think about this requirement?

> But actually the UI code is complicated itself(for console driver itself). We have multiple UI calls for single 4GL statement. So handling the UI requests we can not skip/substitute only single call letting other to be handled as usual because the UI code methods are very coupled one with other.

I'm referring to the code that initializes batch mode here. I understand that the actual honoring of batchmode will require some conditional code in many places. My concern is that it is not clear:

1. Why the batch driver doesn't start out in batch mode.
2. Why the call to activateBatch() must be made at the specific point in the ClientCore processing.

> That'a why I use the term "active" for batch driver. This means the special processing for i/o requests. When inactive we need to pass the control to the underlying console driver layer to provide code consistency and handle redirected cases(when the batch driver works as usual console driver).

I still want to name activateBatch() differently. The call to activateBatch() makes the reader think that it is "unconditionally" shifting into batch mode. But it is testing whether it should shift or not, based on whether there is the batch driver in use or not. The naming is confusing.

> > So, it seems to me that the processing in classes like StreamWrapper that call LogicalTerminal, should instead call something in ErrorManager and the ErrorWriter interface should be extended to provide the necessary support.
>
> In this case the ErrorManager class will contain the methods to identify the batch mode and to stop the client itself. Is it OK to have these client specific methods in ErrorManager/ErrorWriter?

ErrorWriter is exactly the place for it because the point is for instances of this interface to hide the actual client implementation.

The ErrorManager will just delegate these calls to the ErrorWriter instance.

> Need to think what to do with error display in Windows. Currently I've used Swing dialogue to implement this. But in Windows 4GL the dialogue is native. So probably we need to change P2J.DLL adding respective JNI. What do you think about this requirement?

Actually, I think we need to remove all the Swing code and we don't need any JNI support.

In Windows ChUI, any VIEW-AS MESSAGE-BOX will use this same dialog. Is that right?

We are already working on implementing the Windows GUI support. When that is in place, we can make sure that the ChUI implementation uses the same back-end.

#68 - 12/12/2013 08:32 AM - Eugenie Lyzenko

> Actually, I think we need to remove all the Swing code and we don't need any JNI support.
>
> In Windows ChUI, any VIEW-AS MESSAGE-BOX will use this same dialog. Is that right?

Not exactly. In CHUI mode this produces the character based alert box inside character console. In GUI mode this produces Windows native dialog box. In batch mode this produces the "No output destination" error box(as native Windows dialog).

So currently we are planning to use only character messages in console for batch more errors in Windows(the same as for Linux). Correct?

#69 - 12/12/2013 08:53 AM - Greg Shah

> In batch mode this produces the "No output destination" error box(as native Windows dialog).

OK.

> So currently we are planning to use only character messages in console for batch more errors in Windows(the same as for Linux). Correct?

Yes. Please do put TODO comments into the source code to explain in each place where there is a deviation (makes sure to explain what it should be doing there).

#70 - 12/12/2013 10:20 AM - Eugenie Lyzenko

> I'm referring to the code that initializes batch mode here. I understand that the actual honoring of batchmode will require some conditional code in many places. My concern is that it is not clear:

I see.

1. Why the batch driver doesn't start out in batch mode.

The client init code requires the terminal subsystem to be in real console mode(e.g. double buffered terminal output).

2. Why the call to activateBatch() must be made at the specific point in the ClientCore processing.

This is the place just before the calling to the main routine on the server. Last code point where every init processing should be done.

As the option it is now also possible to remove batch mode activation from ClientCore code leaving internal active flag to false. Then the code for UPDATE/SET or other i/o statements will activate the batch special processing.

My idea was to avoid tracking every statement that requires the error to display. Instead I wanted to handle lower level I/O requests when the client is trying to use the missing I/O device. But unfortunately Windows and Linux 4GL has different error display scenario for several statements. So I had to involve the code on the server side for such statements taking into account the underlying OS.

Agree, unconditional OutputManager.activateBatch() call in code letting the reader to suggest this mode will be turned on/off in all cases. What about something like this:

```
...
if (OutputManager.isInBatchMode())
{
   OutputManager.activateBatch()
}
...
```

It this acceptable?

#71 - 12/12/2013 11:10 AM - Greg Shah

The client init code requires the terminal subsystem to be in real console mode(e.g. double buffered terminal output).
This is the place just before the calling to the main routine on the server. Last code point where every init processing should be done.

Please put comments in the code to explain this.

> As the option it is now also possible to remove batch mode activation from ClientCore code leaving internal active flag to false. Then the code for UPDATE/SET or other i/o statements will activate the batch special processing.

No, this is not a better approach.

> It this acceptable?

Yes.

#72 - 12/13/2013 02:53 PM - Eugenie Lyzenko

- File evl_upd20131213a.zip added

The update for the review includes:

1. Merging with the recent code base
2. Changes for notes 1-7 from #65 above
3. Removing Swing usage for Windows client.

#73 - 12/17/2013 07:21 PM - Eugenie Lyzenko

- File evl_upd20131217a.zip added

The next update for review. Includes small fixed discovered in test-cases debugging and code to suppress the batch mode errors for redirected batch mode code. And if there are no objections/notes I guess it is ready enough for regression testing.

#74 - 12/19/2013 06:07 PM - Eugenie Lyzenko

- File evl_upd20131219a.zip added

The update implements the strange 4GL feature founded during debugging batch mode redirected cases. The ENABLE statement generates the error but in case of the redirected output the message is writing to the output file instead of the console window as it is happening for not redirected cases.

To be able to implement this feature the new method displayMessageRedirected() has been added to ErrorManager, ErrorWriter, all writer implementations and LogicalTerminal, ClientExports and ThinClient.

#75 - 12/20/2013 09:14 AM - Greg Shah

Code Review 1219a

1. All EVL** comments need to be removed so that this can be finalized. In general, I prefer if you do not use this in the future. Use the features of "bzr diff" and tools like diff, mgdiff or meld to easily see your changes without these extra comments.

2. The StandardServer call to ErrorManager.initErrorWriter(new ErrorWriterServer()) needs a comment to explain that this call MUST be positioned AFTER the LogicalTerminal is initialized.

3. Please change code like this:

```
      if (sd != null)
      {
         return sd.inBatchMode();
      }
      else
      {
         return false;
      }
```

To the much more condensed form:

```
      return (sd != null) ? sd.inBatchMode() : false;
```

}
You can find this example in OutputManager. This should be the way you write such code in other cases too. For example, see ErrorWriter[Batch|Interactive].isInBatchMode().

4. In ErrorManager, under what conditions will errorWriter == null?

5. What is left to do for this task? Are all features fully implemented? It sounds like you are still debugging the implementation. Is that right?

#76 - 12/20/2013 10:33 AM - Eugenie Lyzenko

1. All EVL** comments need to be removed so that this can be finalized. In general, I prefer if you do not use this in the future. Use the features

of "bzr diff" and tools like diff, mgdiff or meld to easily see your changes without these extra comments.

OK. Of course it should be removed from final version. I just used this to simple identify the new code(on the development stage e.g.).

2. The StandardServer call to ErrorManager.initErrorWriter(new ErrorWriterServer()) needs a comment to explain that this call MUST be positioned AFTER the LogicalTerminal is initialized.

OK.

3. Please change code like this:

if (sd != null) {
return sd.inBatchMode();
}
else {
return false;
}

To the much more condensed form:

return (sd != null) ? sd.inBatchMode() : false;

}
You can find this example in OutputManager. This should be the way you write such code in other cases too. For example, see ErrorWriter[Batch|Interactive].isInBatchMode().

OK.

4. In ErrorManager, under what conditions will errorWriter == null?

I have introduced this checking for cases when something goes wrong to avoid null pointer usage. Normally it should not happen but may be failure in initErrorWriter()? Is this not necessary?

5. What is left to do for this task? Are all features fully implemented? It sounds like you are still debugging the implementation. Is that right?

I think all done now. Yesterday I've debugged the redirected cases to find out possible issues(ENABLE behavior for example).

#77 - 12/20/2013 11:20 AM - Greg Shah

> Normally it should not happen but may be failure in initErrorWriter()? Is this not necessary?

It is good that you do this. I just wanted to make sure that I understand any case where this would actually happen.

#78 - 12/20/2013 01:18 PM - Eugenie Lyzenko

- File evl_upd20131220a.zip added

This update cleans previous notes.

Also I remembered one point under the question. The BatchPrimitives.java class returns 80x24 as terminal size for batch mode for screenWidth() and screenHeight(). The Windows returns some values that can not be interpreted as terminal size(looks like desktop size in character units). But in Linux we have undefined values(?) returning for these calls(WIDTH and HEIGHT).

The key point here is he have to return some correct values here to properly allocate internal cell bitmap. Otherwise the screen driver will fail. Is it OK to leave the return values as 80x24 for this?

#79 - 12/20/2013 06:09 PM - Greg Shah

Code Review 1220a

I am fine with the changes.

> The key point here is he have to return some correct values here to properly allocate internal cell bitmap. Otherwise the screen driver will fail. Is it OK to leave the return values as 80x24 for this?

Yes, this is OK. However, please do add comments to explain how this deviates from both Windows and Linux behavior (explain each one).

After adding the comments, upload that version and start testing.

#80 - 12/20/2013 06:23 PM - Eugenie Lyzenko

- File evl_upd20131220b.zip added

> After adding the comments, upload that version and start testing.

OK. The comments have been added.

#81 - 12/20/2013 06:39 PM - Greg Shah

That is good. Get this tested.

#82 - 12/20/2013 06:41 PM - Eugenie Lyzenko

> That is good. Get this tested.

The testing is on the way.

#83 - 12/21/2013 07:52 AM - Eugenie Lyzenko

- File evl_upd20131221a.zip added

Merging ErrorManager.java with recent code base(10424). Will be used in testing from now.

#84 - 12/21/2013 03:57 PM - Eugenie Lyzenko

- File evl_upd20131221b.zip added

Merging StandardServer.java with recent code base(10425). Will be used in testing from now.

#85 - 12/24/2013 08:35 AM - Eugenie Lyzenko

The testing results summary at this time:
1. The main part is OK.
2. The 3-way CTRL-C tests are passed.
3. The 2-way CTRL-C tests are still in progress because on some step(34-35) we have the following error:

```
...
|        ATT+---------------------- Message -----------------------+061A
|        P.O| An error has occurred attempting to connect to the   |
|        Wil|              selected database system.               |
|           | ---------------------------------------------------- |
...
```

I think this is not P2J error itself, some external issue. May be we reach the maximum simultaneous DB connection?

Continue to get CTRL-C tests passed.

#86 - 12/25/2013 04:30 PM - Eugenie Lyzenko

Greg,

The runtime regression tests are finally all passed. The results can be found in 10425_ed8b6f3_20131225_evl.zip. The CTRL-C are all OK - had to restart multiple times, even some tests separately. But I guess everything is OK.

So I can commit into bzr and distribute evl_upd20131221b.zip.

#87 - 01/04/2014 01:23 PM - Eugenie Lyzenko

Well, I've got the CTRL-C tests fully passed in one round. Take a look at 2014 year results from:
10425_ed8b6f3_20140104_evl.zip

#88 - 01/06/2014 08:53 AM - Greg Shah

I am OK with the changes.

Constantin: do you see any concerns?

If not, then the check in can proceed.

#89 - 01/06/2014 09:18 AM - Constantin Asofiei

Greg Shah wrote:

> Constantin: do you see any concerns?

I don't see any obvious problems in the logic. My only concern was the headless mode for ErrorManager (used by PL/Java), but the headless logic doesn't look to be touched.

#90 - 01/06/2014 10:15 AM - Greg Shah

Go ahead, commit and distribute it.

#91 - 01/06/2014 02:05 PM - Eugenie Lyzenko

evl_upd20131221b.zip did pass the regression testing and committed to bzr as 10426.

#92 - 01/06/2014 02:36 PM - Greg Shah

- % Done changed from 0 to 100

- Status changed from New to Closed

#93 - 01/14/2014 06:05 AM - Constantin Asofiei

There is a memory leak in the TypeAhead.taBuf list, when batch mode is activated. On linux, when the BatchDriver.readKey API is called by TypeAhead$KeyReader.run, they F1 is returned always (without blocking); as the TypeAhead$KeyReader.run has a while(true) loop which was built with the "it will block for user input" case in mind, this will act as an infinite loop and will leak the F1 key to the TypeAhead.taBuf list.

This is also shown by the following test:

```
readkey.
output to foo.txt.
display last-key.
output close.
```

- if ran under linux with bpro -p a.p the content of foo.txt will be -2.
- if ran under linux with pro -b -p a.p will output:

```
** Attempt to read with no current source of input. (513)
```

As a side note, we have different behaviour, even if both cases are "batch mode".

My conclusions are:

1. the error logic in BatchDriver.readkey should not have been there. I think the proper location is in ThinClient.readkey - as these errors are shown only when READKEY is called, right?
2. if batch mode is enabled, then the KeyReader thread should not be started at all (as blocking for user input is impossible, right?). TypeAhead.dequeueEvent will always return null, as there will be no keys to read.
3. for appserver case at least, I found that I need to set the client:mode:batch=true to the client startup command and EnvironmentOps.setBatchMode(true) called on server side. We should not be required to specify batch mode for both the client-side and server-side (for the same P2J client); we should determine a single location for this setting and interrogate that both on client and server side. More, in OutputManager.initErrorWriter (which is called from ThinClient.initializePrep), EnvironmentOps.isBatchMode() will always return false. EnvironmentOps.isBatchMode() will never interrogate the server-side, as it was meant to be called only on server-side, not on client-side.

#94 - 01/14/2014 08:27 AM - Greg Shah

> if batch mode is enabled, then the KeyReader thread should not be started at all (as blocking for user input is impossible, right?). TypeAhead.dequeueEvent will always return null, as there will be no keys to read.

I think this is generally correct. However, I wonder what the 4GL does when you "manually" send it the CTRL-C via kill -INT <pid>?

This is a scenario we must handle the same way. Today, our CTRL-C processing is very dependent upon the KeyReader thread, so we may have to accommodate this case. It would be a way for 4GL devs to use scripting to cause a 4GL batch program to exit. As such, it seems like it would be a very useful technique and one we must support.

#95 - 01/14/2014 08:31 AM - Eugenie Lyzenko

> if ran under linux with pro -b -p a.p will output:

```
> ** Attempt to read with no current source of input. (513)
```

The correct command for batch mode in Linux is bpro, not pro. And in Linux there should be no error messages for batch mode with your test.

#96 - 01/14/2014 08:54 AM - Constantin Asofiei

Greg Shah wrote:

> However, I wonder what the 4GL does when you "manually" send it the CTRL-C via kill -INT <pid>?

I've used this test to check if any conditions are raised:

```
def var ch as char.
def var i as int.

output to foo.txt.
do on error undo,leave:
  do on stop undo, leave:
     do on quit undo, leave:
        do on endkey undo,leave:
            do i = 1 to 100000000:
              ch = string(i).
            end.
            message "endkey : not".
        end.
        message "quit: not".
     end.
     message "stop: not".
  end.
  message "error: not".
end.

message "no-conditions".

output close.
```

Tested with both bpro and pro -b:

1. with bpro, the program starts running in background; kill -INT will not terminate it (no condition is raised). kill -SIGKILL will terminate it
2. with pro -b, the program starts, but not in background; pressing CTRL-C or kill -INT will raise the STOP condition in the program.

Eugenie Lyzenko wrote:

> The correct command for batch mode in Linux is bpro, not pro

Is this officially documented somewhere? Because even if it is the official command, this will not stop someone from using pro -b.

Greg: do we go ahead and assume the code is ran using bpro? Because bpro and pro -b have different behaviour; and considering your note that 4GL devs might "use scripting to cause a 4GL batch program to exit", I suspect they would use pro -b, and not bpro.

#97 - 01/14/2014 08:57 AM - Eugenie Lyzenko

> The correct command for batch mode in Linux is bpro, not pro

> Is this officially documented somewhere? Because even if it is the official command, this will not stop someone from using pro -b.

Yes, this is official command. The document "OpenEdge Getting Started: Installation and Configuration", page 324.

#98 - 01/14/2014 09:04 AM - Eugenie Lyzenko

> There is a memory leak in the TypeAhead.taBuf list...

Explain what do you mean telling "memory leak". Another word what exactly is the problem we need to fix here. Does the client java session eat memory from system? Something else?

And BTW in Windows we have to display error and stop application for readkey. So I doubt we have to suppress KeyReader thread.

#99 - 01/14/2014 09:13 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> There is a memory leak in the TypeAhead.taBuf list...

> Explain what do you mean telling "memory leak". Another word what exactly is the problem we need to fix here. Does the client java session eat

memory from system? Something else?

On each iteration of the while(true) loop in TypeAhead$Keyreader.run:403, there will be a call to OutputManager.instance().readKey(). In batch mode, this will return immediately and the returned value (F1) will be added to the TypeAhead.taBuf list. With rev 10436, this loop is an infinite loop in batch mode; assume you've just started a client in batch mode and you have a breakpoint right at the beginning of the program's execute method (to simulate a very long running batch process). On P2J client-side, the Keyreader thread keeps "reading" the F1 value indefinitely; which results in, as you say, "the client java session eat memory from system".

And BTW in Windows we have to display error and stop application for readkey. So I doubt we have to suppress KeyReader thread.

I understand these errors need to be displayed on Windows. But wouldn't be more appropriate to implement these errors at the implementation of the READKEY function, on client-side (ThinClient.readkey)? As I understand, only a call of READKEY will display the error ~~on~~ and stop the application on Windows.

#100 - 01/14/2014 09:28 AM - Greg Shah

The correct command for batch mode in Linux is bpro, not pro

Is this officially documented somewhere? Because even if it is the official command, this will not stop someone from using pro -b.

Yes, this is official command. The document "OpenEdge Getting Started: Installation and Configuration", page 324.

Actually, this is not quite right. From the "OpenEdge Deployment: Startup Command and Parameter Reference", v11.0, page 43:

```
BPRO command
Starts a single-user OpenEdge client session in batch or background mode.
```

This doesn't help too much. But further down the page there is this:

```
- For UNIX, the BPRO command runs the following executable:

    _progres -1 -b &

  The & character causes an OpenEdge batch session to run in the background, and
  returns control to the terminal once the session starts. When the & character is not
  present (for example, when you simply use the (-b) startup parameter),
  OpenEdge initiates a batch session in the foreground without terminal interaction
```

```
  and control does not return to the terminal until the session completes.

- On UNIX and in Windows, you can redirect batch job input and output with the <
  and > redirection symbols. You also can use the pipe symbol (|) to put an
  OpenEdge batch run in a command pipeline.
```

This is very instructive. Although bpro does also implement batch mode (-b option), more importantly it starts the session in BACKGROUND mode which has very specific implications in regard to the shell/process environment AND which explains why it doesn't respond to the SIGINT since it has no foreground terminal.

The key question: can we duplicate this behavior using the & directly from the shell?

In regard to batch mode itself, ANY 4GL session that has the -b option on the command line will be in batch mode. We MUST support this as this is a common way that batch mode is used. From page 89 of the same book:

```
Batch (-b)

Use Batch (-b) to initiate a batch session, with no terminal interaction.
When ABL (Advanced Business Language) procedures run in batch mode, the ABL
Virtual Machine (AVM) does not display any messages produced by those procedures
on the terminal screen. However, AVM error messages are displayed to the terminal
screen. To suppress these messages, redirect standard output as shown in this Windows example:

prowin32.exe -p test.p -b > nul
```

#101 - 01/14/2014 12:09 PM - Eugenie Lyzenko

> The key question: can we duplicate this behavior using the & directly from the shell?

May be this is not required for our case(see suggestions below)

> In regard to batch mode itself, ANY 4GL session that has the -b option on the command line will be in batch mode. We MUST support this as this is a common way that batch mode is used.

Do we need to support both foreground and background mode?

Because the & option is used with -b option I think we do not need the special processing for "&" option. Instead we can use two modes for batch processing in client.sh script:

```
swing="client:driver:type=chui_batch client:driver:mode=background"
```

or

```
swing="client:driver:type=chui_batch client:driver:mode=foreground"
```

with default value as foreground for line:

```
swing="client:driver:type=chui_batch"
```

Anyway we will need to introduce new member flag like

```
boolean modeForeground = true;
```

into the BatchPrimitive.java class to separate the behavior when there is a difference between modes.

#102 - 01/14/2014 06:02 PM - Greg Shah

> Do we need to support both foreground and background mode?

Yes.

> Because the & option is used with -b option I think we do not need the special processing for "&" option. Instead we can use two modes for batch processing in client.sh script:

The use of & is nothing to do with the 4GL. It is not an option and is not actually passed to the _progres executable. Instead, it is a feature of the operating system shell (on Linux, this is usually /bin/bash). This is a UNIX/Linux thing and does not occur on Windows.

My concern with your suggested approach is that we would have to find every place that is different when using the & and implement that difference manually in our code. And there may be some things that can only be provided at the OS/shell level and cannot be properly duplicated in Java.

#103 - 01/14/2014 06:03 PM - Greg Shah

It is important that you make sure you re-run all your tests using the pro -b since our current code may have some differences from how your first batch mode implementation works (since your previous testing was based on using bpro).

#104 - 01/14/2014 06:11 PM - Eugenie Lyzenko

- File evl_upd20140114a.zip added

> It is important that you make sure you re-run all your tests using the pro -b since our current code may have some differences from how your first batch mode implementation works (since your previous testing was based on using bpro).

OK. I'll verify the tests. This is the suggested fix(not including foreground/background differences) for you to review. It bypass the regular key reading logic and returns always -2(EOF) key code(event) as it is happening in 4GL. Type ahead buffer does not start the reading thread for batch mode.

#105 - 01/14/2014 06:49 PM - Eugenie Lyzenko

> My concern with your suggested approach is that we would have to find every place that is different when using the & and implement that difference manually in our code. And there may be some things that can only be provided at the OS/shell level and cannot be properly duplicated in Java.

Unfortunately(this is not I like but we had) we already do this separation making differences between reactions for batch mode errors for Linux and Windows OS.

#106 - 01/15/2014 02:21 AM - Constantin Asofiei

The good news is that I think pro -b & behaves the same as pro -b (kill -INT will raise a STOP condition in the program). Thus, we need to differentiate only between bpro and pro -b and implement the behaviour accordingly.

The approach using client:driver:mode=foreground and client:driver:mode=background I think can be translated better into a client:driver:background=true flag - thus, this will be specified only if the client needs to be ran in "background" mode.

#107 - 01/15/2014 12:40 PM - Eugenie Lyzenko

- File evl_upd20140115a.zip added

The drop adds foreground/background differences introduced via client.sh script "client:driver:background=true|false" option. Some fixes was included also when running testcases for b/f modes to reflect the 4GL behavior in these modes. This is drop for you to review, I guess it is ready to be tested for regressions if there are no objections or other notes.

#108 - 01/15/2014 12:56 PM - Constantin Asofiei

Eugenie: BatchDriver.init I don't think it should output the progress logo; it is enough to output the messages after the logo.

I'll check tomorrow how the appserver connection behaves.

#109 - 01/15/2014 12:59 PM - Eugenie Lyzenko

Eugenie: BatchDriver.init I don't think it should output the progress logo; it is enough to output the messages after the logo.

This is exactly what I see running the command pro -p some_program.p -b instead of bpro ...

#110 - 01/15/2014 01:07 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Eugenie: BatchDriver.init I don't think it should output the progress logo; it is enough to output the messages after the logo.
>
> This is exactly what I see running the command pro -p some_program.p -b instead of bpro ...

Yes, but I think that is emitted by proprietary 4GL code owned by progress (or at least the logo might be copyrighted). Greg, let us know your thoughts.

#111 - 01/15/2014 01:16 PM - Greg Shah

Code Review 0115a

1. Copyright ranges need to be updated to 2014 for all files.

2. We DEFINITELY CANNOT include that Progress logo screen! I understand and approve of your instinct to make things exactly compatible. But in this case compatibility is not an issue and it could cause people to be confused about who owns the P2J technology. Please remove that.

3. As I have mentioned before, please code things like this:

```
      if (some_test)
      {
         return option1;
      }
      else
      {
         return option2;
      }
```

as this:

```
      return (some_test) ? option1 : option2;
```

You can see examples of this in ErrorWriterBatch.

4. Have you tested sending SIGINT to the resulting batch process? I think it won't raise a STOP condition but it should. This is related to the KeyReader thread change. If I am right, then we need to come up with a solution.

#112 - 01/15/2014 01:35 PM - Eugenie Lyzenko

1. Copyright ranges need to be updated to 2014 for all files.

Done.

2. We DEFINITELY CANNOT include that Progress logo screen! I understand and approve of your instinct to make things exactly compatible. But in this case compatibility is not an issue and it could cause people to be confused about who owns the P2J technology. Please remove that.

OK. I'm leaving for background mode:

```
Batch processing will be performed using:
OpenEdge Release ...
```

and for foreground mode:

```
OpenEdge Release ...
```

Is this OK?

3. As I have mentioned before, please code things like this:

```
> if (some_test)
> {
> return option1;
> }
> else
> {
> return option2;
> }
>
```

OK.

4. Have you tested sending SIGINT to the resulting batch process? I think it won't raise a STOP condition but it should. This is related to the KeyReader thread change. If I am right, then we need to come up with a solution.

No, I have not tested. You mean the reaction to CTRL-C that should restart the client session?

#113 - 01/15/2014 01:58 PM - Greg Shah

> OK. I'm leaving for background mode:
>
> Batch processing will be performed using:
> OpenEdge Release ...
>
> and for foreground mode:
>
> OpenEdge Release ...
>
> Is this OK?

No, I think it is better to remove these. Put comments into the code that describes that these things were intentionally removed. I don't want anyone to get the idea that P2J is OpenEdge.

> 4. Have you tested sending SIGINT to the resulting batch process? I think it won't raise a STOP condition but it should. This is related to the KeyReader thread change. If I am right, then we need to come up with a solution.
>
> No, I have not tested. You mean the reaction to CTRL-C that should restart the client session?

Yes, except since there is no user input in these modes, the way to pass a CTRL-C to the process is by using kill -SIGINT <pid>. If it properly generates a STOP condition in the converted code, then it is working.

#114 - 01/15/2014 04:21 PM - Eugenie Lyzenko

> Yes, except since there is no user input in these modes, the way to pass a CTRL-C to the process is by using kill -SIGINT <pid>. If it properly generates a STOP condition in the converted code, then it is working.

It is strange but in my system kill -SIGINT <pid> terminates the regular console application(when we can use CTRL-C to restart). What do I wrong? I'm taking the pid from ps aux | grep "java", choose ClientDriver and execute: kill -s SIGINT <pid> or kill -SIGINT <pid> or kill -s INT <pid>. The same result - the client session silently terminates. Do I need to send SIGINT to the key reader thread instead of whole client?

#115 - 01/15/2014 04:41 PM - Greg Shah

I'm not sure, but I do wonder if it has something to do with our signal handling approach.

The sigaction() calls that used to be in the init() method of libTerminal.so were moved into init() in process_linux.c. But I don't think we actually call that init() code until we actuall try to call any method in LaunchManager, which is only done when we start a child process. That means that for normal usage, the signal handling configuration is never put in place at process start since we stopped using libTerminal. I suspect this may be part of the problem.

But that likely will just make the process ignore the signal (not die).

I think we will still have a problem of generating the STOP condition.

#116 - 01/15/2014 05:08 PM - Eugenie Lyzenko

- File evl_upd20140115b.zip added

OK. I'll need to dig more to clarify the STOP condition generation.

For now I've uploaded the changes clearing all other notes with batch mode fix(key reader thread is not starting in this drop until I have more details).

The testcases I'm using to debug STOP condition in batch are:

```
def var ch as char.
def var i as int.

output to "batch_test20.log".
output close.

output to "batch_test20.log" append.
message "program started".

do on error undo,leave:
  do on stop undo, leave:
     do on quit undo, leave:
        do on endkey undo,leave:
            do i = 1 to 100000000:
              ch = string(i).
            end.
            message "endkey : not".
        end.
        message "quit: not".
     end.
     message "stop: not".
  end.
  message "error: not".
end.

message "no-conditions".

output close.
```

and

```
def var ch as char.
def var i as int.
def var j as int.

output to "batch_test21.log".
message "program started".

do on stop undo, leave:
   do i = 1 to 100000000:
      do j = 1 to 100000000:
         ch = string(i).
      end.
   end.
   message "stop: not".
end.

message "no-conditions".

output close.
```

#117 - 01/16/2014 03:18 AM - Constantin Asofiei

Eugenie, two notes:

1. check how kill -INT is treated for normal start in 4GL (non-batch). Double-check this behaviour with non-batch clients in P2J - I suspect in 4GL we should have a STOP condition raised; if so, this should happen in P2J also.
2. if I recall correctly, P2J has problems interrupting a long-running loop with no UI or DB interaction. Just something to keep in mind when testing.

#118 - 01/16/2014 04:01 AM - Constantin Asofiei

About the update:

- the appserver connection is working properly.
- ThinClient.isBatchInBackground - there is a warning related to outMgr.isBatchInBackground() - this is a static method, and should be accessed via OutputManager.isBatchInBackground().

#119 - 01/16/2014 06:37 AM - Greg Shah

Code Review 0115b

1. Please put this comment in BatchDriver.init()

```
// the 4GL outputs a Progress logo, copyright notice and version at the startup of batch mode
// when run in the foreground (and just the version when run in the background); we are
// deliberately NOT duplicating these outputs for these reasons:
// 1. they have no value for application/user processing or functionality
// 2. generating that output would confuse people as to who owns this technology
// 3. these values changes from version to version anyway, so it is very unlikely that an app
//    would depend upon the output
```

2. The BatchDriver copyright year range probably should be 2013-2014 instead of 2010-2014, right?

#120 - 01/16/2014 09:05 AM - Eugenie Lyzenko

> Eugenie, two notes:
>
> check how kill -INT is treated for normal start in 4GL (non-batch). Double-check this behaviour with non-batch clients in P2J - I suspect in 4GL we should have a STOP condition raised; if so, this should happen in P2J also.

I have tested the simple app with one fill-in in UI mode. The result is the same - kill -INT just terminates the client session to the command prompt, not restart the client as it is expected.

#121 - 01/16/2014 09:10 AM - Eugenie Lyzenko

- File evl_upd20140116a.zip added

The drop fixes the today notes.

#122 - 01/16/2014 10:25 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> I have tested the simple app with one fill-in in UI mode. The result is the same - kill -INT just terminates the client session to the command prompt, not restart the client as it is expected.

I'm not sure I understand what you've tested - your test was in P2J or 4GL? On lindev01, I've tried an UPDATE statement and when the program is started with pro -p, on CTRL-C will restart it. Sending kill -INT to it will restart it too (the program will not end). Checking with programs like in note 116, a STOP condition is raised in this case too.

#123 - 01/16/2014 10:36 AM - Eugenie Lyzenko

> I'm not sure I understand what you've tested - your test was in P2J or 4GL?

OK. I have tested the our P2J client implementation for UI console mode. And kill behaves not the same as manual CTRL-C.

#124 - 01/16/2014 10:41 AM - Greg Shah

> I have tested the our P2J client implementation for UI console mode. And kill behaves not the same as manual CTRL-C.

OK, that means we have a problem to fix.

Try this: temporarily re-enable the KeyReader thread and re-test. Does the kill -INT generate a CTRL-C "key" that is read by that thread?

#125 - 01/16/2014 06:14 PM - Eugenie Lyzenko

> OK, that means we have a problem to fix.
>
> Try this: temporarily re-enable the KeyReader thread and re-test. Does the kill -INT generate a CTRL-C "key" that is read by that thread?

No, for regular CHUI test(KeyReader is active) kill -INT does not generate CTRL-C

Some findings.

1. SIGINT causes the SilentUnwindException on the server side StandardServer.invoke() with reason - connection lost.
2. It is happening after client terminates.
3. Java VM on the client side consider this signal as command to terminate because we do not install the signal handlers on the client side.

I think the main issue here is CTRC-C and SIGINT are not exactly the same. CTRL-C is one of the reason for SIGINT. We intercept only key events on the client side in TypeAhead class. And OutputManager.instance().readKey() which is native getch() from NCURSES for CHUI console returns -1 when SIGINT is receiving. May be we need to install signal handler into libp2j.so library by sigaction() call to intercept SIGINT and transform it to keyboard event CTRL-C. However this can work only if java.exe is alive at the moment libp2j.so is receiving SIGINT I think.

What do you think about this: http://www.dclausen.net/javahacks/signal.html ? A kind of Java level signal processing integrated into JVM?

#126 - 01/16/2014 06:24 PM - Greg Shah

Before you tested, did you fix this issue I previously mentioned?

> The sigaction() calls that used to be in the init() method of libTerminal.so were moved into init() in process_linux.c. But I don't think we actually call that init() code until we actuall try to call any method in LaunchManager, which is only done when we start a child process. That means that for normal usage, the signal handling configuration is never put in place at process start since we stopped using libTerminal. I suspect this may be part of the problem.

Without this, the client will definitely exit upon getting the SIGINT.

#127 - 01/16/2014 06:52 PM - Eugenie Lyzenko

> Before you tested, did you fix this issue I previously mentioned?
>
> > The sigaction() calls that used to be in the init() method of libTerminal.so were moved into init() in process_linux.c. But I don't think we actually call that init() code until we actuall try to call any method in LaunchManager, which is only done when we start a child process. That means that for normal usage, the signal handling configuration is never put in place at process start since we stopped using libTerminal. I suspect this may be part of the problem.
>
> Without this, the client will definitely exit upon getting the SIGINT.

OK. Now I have moved the code from init() to initConsole() to test. The situation is better. The client is not terminating, the INT signal is just ignoring(the CTRL-C again is not passing to client, nor STOP condition to the server).

So we need to install our own handler for INT into libp2j.so. This should help.

#128 - 01/16/2014 08:17 PM - Greg Shah

We can't use the Sun signal handling support otherwise this would limit our customer's usage of alternative JVMs.

> So we need to install our own handler for INT into libp2j.so. This should help.

Yes. The sigaction() usage needs to be executed every time the client starts (instead of only when a child process launches). But for SIGINT, we must do something different.

I think that we can have a Java thread that calls into libp2j and then use sigwait() to block waiting for SIGINT. When it gets the signal, it should probably block or ignore the SIGINT, return to Java, trigger the ThinClient.watchCtrlC() and then call back down to native code to sigwait() again.

Things to consider:

1. We can't let a SIGINT kill the process if it is delivered while the thread is calling ThinClient.watchCtrlC(). That would be a race condition where the delivery of 2 SIGINT very quickly could result in the process being killed.

2. We have to make sure that the thread will be properly cleaned up when the process is supposed to exit. Normally, we do this by making a Java thread a "daemon" thread. But I worry that the native blocking may break this behavior.

3. This new feature would always be enabled on Linux/UNIX since it isn't just used for batch or background modes.

4. What about Windows?

#129 - 01/17/2014 06:58 AM - Eugenie Lyzenko

> Yes. The sigaction() usage needs to be executed every time the client starts (instead of only when a child process launches). But for SIGINT, we must do something different.
>
> I think that we can have a Java thread that calls into libp2j and then use sigwait() to block waiting for SIGINT. When it gets the signal, it should probably block or ignore the SIGINT, return to Java, trigger the ThinClient.watchCtrlC() and then call back down to native code to sigwait() again.
>
> Things to consider:
>
> 1. We can't let a SIGINT kill the process if it is delivered while the thread is calling ThinClient.watchCtrlC(). That would be a race condition where the delivery of 2 SIGINT very quickly could result in the process being killed.
>
> 2. We have to make sure that the thread will be properly cleaned up when the process is supposed to exit. Normally, we do this by making a Java thread a "daemon" thread. But I worry that the native blocking may break this behavior.
>
> 3. This new feature would always be enabled on Linux/UNIX since it isn't just used for batch or background modes.

OK. Agree with this approach. I have tested the sigaction() and signal handler moving the related code from ProceeLauncher.init() native function to the console driver initialization. The new handler just emits CTRL-C by calling ungetch(3). The result - CHUI application restarts in both manual CTRL-C and SIGINT cases. This was dark force solution for testing.

And moreover the approach you are talking about will not depend on whether we need to start the key reader thread(in CHUI mode) or not(in batch mode).

> 4. What about Windows?

I'll check this later in details but Windows certainly has the specifics in this area. Another things to discuss is about the signal processing init code for all required signals to special process. We load "p2j" native library in several places and every time we need to make sure the signal init is called after library loading. What about to combine and separate these two calls into single place, say inside the com.goldencode.util.PlatformHelper. We can create static call PlatformHelper.loadNativeP2JLibrary() with just two things:

```
System.loadLibrary("p2j");
initSignalsProcessing();
```

The initSignalsProcessing() will be the native call from "p2j" that performs handler installing and sigaction() for the rest of the signals. For every OS we create the different init native code to take into account the OS specifics in signal processing. Also the loadNativeP2JLibrary() should verify the real signal initialization be called only once to avoid multiple signals redefining. What do you think about this plan?

#130 - 01/17/2014 08:40 AM - Greg Shah

> Another things to discuss is about the signal processing init code for all required signals to special process. We load "p2j" native library in several places and every time we need to make sure the signal init is called after library loading.

Yes.

Actually, to properly support the behavior we need to make sure that every time a P2J client process starts, that the library is loaded and this init is done.

> What about to combine and separate these two calls into single place,

Yes, good idea.

> say inside the com.goldencode.util.PlatformHelper.

Things in com.goldencode.util.* are not supposed to have any "knowledge" or references to P2J at all. These are places for generic helpers/utilities.

I think it is better to put this into com.goldencode.p2j.main.ClientCore which is code that is always involved when the client is started (batch, interactive chui, interactive gui or appserver).

> The initSignalsProcessing() will be the native call from "p2j" that performs handler installing and sigaction() for the rest of the signals.

Instead of initSignalsProcessing(), let's call it processInit(). We may have other non-signal logic to put in there later.

From processInit(), you can call a platform-specific function.

> Also the loadNativeP2JLibrary() should verify the real signal initialization be called only once to avoid multiple signals redefining.

Yes, just by placing it properly in ClientCore, you should achieve this.

#131 - 01/17/2014 09:47 AM - Constantin Asofiei

Greg, a note: if there is a chance all the kill -INT behaviour will not be ready until next friday, we should consider releasing an intermediate update (withtout the KILL -INT fix). The mem leak and the status message related fixes are needed for appserver to work in batch mode.

#132 - 01/17/2014 10:09 AM - Greg Shah

> we should consider releasing an intermediate update (withtout the KILL -INT fix)

Agreed.

Eugenie: please work toward having this work done (and regression tested and checked in) by end of day next Friday the 24th. If you believe that is not possible, let us know and we will go forward with the intermediate update.

#133 - 01/17/2014 01:17 PM - Eugenie Lyzenko

> Eugenie: please work toward having this work done (and regression tested and checked in) by end of day next Friday the 24th. If you believe that is not possible, let us know and we will go forward with the intermediate update.

OK. I guess we have the opportunity to be in time.

#134 - 01/17/2014 05:32 PM - Eugenie Lyzenko

Greg,

Is it OK to name the native and OS dependent init modules as initialize.(c|h) and initialize_(linux|win).c or may there are special considerations here?

#135 - 01/17/2014 06:15 PM - Greg Shah

iniitalize_ would be fine. But let's just call it init_ in all cases to keep the filenames shorter.

#136 - 01/17/2014 07:26 PM - Eugenie Lyzenko

- File evl_upd20140117a.zip added

> But let's just call it init_ in all cases to keep the filenames shorter.

OK. This is the prepared base for approach implementation. Just for you to know what is going on here. The EVL tags will certainly be removed from final version, I left them to simplify separate the new code.

#137 - 01/19/2014 05:44 PM - Eugenie Lyzenko

4. What about Windows?

In Windows we do not have sigwait(). So another approach might be acceptable:

1. Install our own handler for SIGINT by signal() call.
2. Inside the new handler post the event semaphore every time SIGINT is arriving.
3. In the function we call from Java waits for event semaphore to be posted.

For the time of processing next SIGINT event we ignore any possible incoming SIGINT's.

#138 - 01/20/2014 09:40 AM - Greg Shah

I haven't gotten to the code review yet. But your approach for Windows is good.

#139 - 01/20/2014 03:44 PM - Eugenie Lyzenko

- File evl_upd20140120a.zip added

This is implementation candidate for Linux SIGINT interceptor.

The Java part consist of the new class SigintWaiter is running from ThinClient.

Unfortunately for now the usage of the sigwait() has no positive results. If the signal was blocked or ignored before sigwait() the sigwait() does not return control to the native caller. If we neither block nor ignore the signal - the client just terminates. So for now I'm using the handler function redefinition. The handler set the internal flag indicating we got the signal. All further signals will be ignored until handler is unlocked(driven from Java code).

This is not refined version, missing Javadocs and so on, just for preliminary review as potential working solution(it is really working). Continue investigation and implementing version for Windows.

#140 - 01/20/2014 05:28 PM - Eugenie Lyzenko

Finding. The folowing code works fine in standalone application but does not work in library(like libp2j):

```
...
sigemptyset(&signalSet);
sigaddset(&signalSet, SIGINT);
```

```
sigprocmask(SIG_BLOCK, &signalSet, NULL);
err = sigwait(&signalSet, &sig);
...
```

Not depending whether we use daemon Java caller thread or not.

#141 - 01/20/2014 08:15 PM - Eugenie Lyzenko

- File evl_upd20140120b.zip added

This update adds the Windows implementation for SIGINT handler. Works in Windows 7 64-bit in virtual machine. Continue testing and investigation.

#142 - 01/21/2014 08:45 AM - Greg Shah

Code Review 0120b

1. We don't need calls to the ClientCore.loadNativeP2JLibrary() from many different files. We only need 1 call to that function. The call can be made from near the top of the ClientCore.start(BootstrapConfig, boolean, boolean, String[]). Just make the call from outside of the while loop, otherwise the call could occur more than once for the process.

2. Change the name loadNativeP2JLibrary() to loadNativeLibrary() and make it private and synchronized.

3. I don't want to use polling. We can properly block the Java daemon thread in native code using event semaphores (Windows) and pthread condition variables (Linux/UNIX - see pthread_cond_wait()/pthread_cond_signal()). Polling is too costly in CPU and makes us less responsive (it takes on average a quarter second to respond to the CTRL-C).

4. The SigintWaiter.watcher member should be private.

5. I think the signals processing should be in native files named signal.c/signal.h/signal_linux.c/signal_windows.c. Leave the init.c/init.h for processInit(), but for now they will only call initSignalsProcessing() which will be in signal.c. Remember that the "common" files have the JNI processing and the platform-specific files have the minimum non-JNI helpers needed to provide services to the common code.

6. The Windows code for initializing signals aborts the process with exit(-1). The Java code should be notified about the problem so that some kind of diagnostic can be logged.

7. Remove the unnecessary commented includes in init.h.

8. build.xml and ThinClient need history entries.

9. Please use bzr diff instead of using the // EVL comments. Bazaar will give you an exact list of all changes you have made. And you can always use bzr revert to remove your changes. So you really don't need those extra comments. Using them just makes extra work later. And it makes the code harder to review for others.

#143 - 01/21/2014 08:47 AM - Greg Shah

The folowing code works fine in standalone application but does not work in library(like libp2j):

I have not checked this, but I suspect that the signal handling may only be delivered on the "main" thread of the process.

#144 - 01/21/2014 08:48 AM - Greg Shah

I am fine with the use of a semaphore to block waiting for the signal. As noted above, this can be implemented the same way for both Windows and for Linux/UNIX and it is a clean solution. It is a good replacement for sigwait().

#145 - 01/21/2014 04:20 PM - Eugenie Lyzenko

Almost all is ready. One point to discuss:
We can not use signal.h as local header file for our library call. It conflicts with GNU-C signal.h header. What about signals.h instead?

#146 - 01/21/2014 04:52 PM - Greg Shah

OK, it makes sense. Change all the names to be signals* instead of signal*.

#147 - 01/21/2014 05:58 PM - Eugenie Lyzenko

- File evl_upd20140121a.zip added

This is the release candidate for review. Works for 64-bit Windows. Need to check with 32-bit, later today.

#148 - 01/22/2014 08:41 AM - Greg Shah

Code Review 0121a

1. After you use throwException() in native code, please don't use exit(-1). Just use return. See signals_win.c and signals_linux.c.

2. I'm surprised that we don't have to link with additional libraries for pthreads on Linux or the event semaphore on Windows.

3. The biggest issue that I see here is that when the SigintWaiter.shutdown() is called, it will not unblock the waiting thread. So that thread will stay waiting until a SIGINT arrives. There should be a native call that triggers the condition var/event sem to unblock the waiting thread. Then the code in SigintWaiter.run() should check the running flag again before triggering watchCtrlC().

4. Does the BatchDriver still need to import com.goldencode.p2j.main.*?

5. Don't import individual classes unless there is a conflict requiring that technique. So:

SigintWaiter import com.goldencode.p2j.main.ClientCore; should be import com.goldencode.p2j.main.*;.
FileChecker import com.goldencode.util.PlatformHelper; should be import com.goldencode.util.*;.

6. SigintWaiter line 70 is missing a space in if(waitForSigint() != -1).

7. makefile is missing a history entry.

#149 - 01/22/2014 09:14 AM - Eugenie Lyzenko

2. I'm surprised that we don't have to link with additional libraries for pthreads on Linux or the event semaphore on Windows.

In Windows it should link to Kernel32.(lib|dll).

#150 - 01/22/2014 10:49 AM - Eugenie Lyzenko

- File evl_upd20140122a.zip added

The update clears the notes for previous code review and merges with the recent codebase from bzr.

#151 - 01/22/2014 11:27 AM - Greg Shah

Code Review 0122a

It looks good. Get it regression tested.

#152 - 01/23/2014 02:08 PM - Eugenie Lyzenko

Looks like the main part is OK. Continue testing with CTRL-C one.

#153 - 01/24/2014 09:29 AM - Eugenie Lyzenko

- File evl_upd20140124a.zip added

The code merge with the 10447. Looks like there are no regressions found.

Do I need to commit and distribute?

#154 - 01/24/2014 09:53 AM - Greg Shah

Code Review 0124a

All the merges look good. Considering that the only tricky merging was in the build scripts, I think your previous regression testing should be sufficient here. As you noted, you passed both the main portion and the CTRL-C tests (which were especially important in light of your changes).

Please do check this in and distribute it.

#155 - 01/24/2014 10:20 AM - Eugenie Lyzenko

The testing results are in:
10446_f202309_20140124_evl.zip
Taking into account all sessions the testing has been passed. So I'm going to distribute the update.

#156 - 01/24/2014 11:53 AM - Greg Shah

This was committed as bzr revision 10449.

#157 - 02/10/2014 04:16 PM - Eugenie Lyzenko

- File testcases_20140210a.zip added

Testcases have been added.

#158 - 02/11/2014 06:15 AM - Constantin Asofiei

- File run_batch.p added

- File batch_logs.zip added

Eugenie, I've ran the attached program (which executes all your tests except batch_test0.p) using the pro -b -p run_batch.p > run_batch.log < batch_test.ini command (both standard input and output are redirected) and they all executed without stopping the batch session. The attached logs were created.

I guess "background" mode was intended only for standard output redirection, right? Because in P2J, even if I set "background" mode on, ThinClient.stopBatchSession gets called.

#159 - 02/11/2014 06:39 AM - Constantin Asofiei

- File ca_srv2.server.log added

This is the log after running run_batch.p on windev01 within an appserver. Same as input redirection on linux: all tests execute, but with different behaviour. The agent is not terminated.

I think the conclusion is that we need to explicitly mark P2J clients as appserver agents, for all of this to work.

Warning: you have some long-running loops in tests batch_test20.p and batch_test21.p. Running these on windev01 will starve the system memory and block your session until the program terminates. You need to reduce these loops to something small, to run them on windev01.

#160 - 02/13/2014 04:40 AM - Constantin Asofiei

Eugenie: the modal message boxes are not supported by background mode. On linux pro -b -p with a MESSAGE ... VIEW-AS ALERT-BOX writes the data to standard output. I think you have these cases to check, for all the I/O statements:

- both windows and linux
- all the pro -b, bpro, and prowin32 -b ways of starting a batch program
- the appserver agents (I know currently this can be tested on 4GL only on windows, but we should assume the behaviour is for linux too - do not restrict this to the Windows platform).

To run a program using an appserver, use this test (on windev01):

```
def var h as handle.
create server h.
message h:connect("-S 2223 -DirectConnect -H localhost -sessionModel Session-Free").
run <program-name> on server h.
message h:disconnect().
```

The home folder for the appserver on windev01 is D:\temp\oe102bworking\ca\1608\batch which contains all the batch_test* programs; you can change/add any program as you wish, I'm not using this appserver or the folder. In the 4GL snippet above, you can change <program-name> with run_batch.p or any other program from that folder. To check the log with STDOUT output, see the D:\temp\oe102bworking\ca_srv2.server.log file.

If you have any questions, do let me know.

#161 - 02/13/2014 04:52 AM - Constantin Asofiei

LE: the all the pro -b, bpro, and prowin32 -b ways of starting a batch program part needs to be expanded with (for each command):

- the output and input redirection cases:
  - only input redirection
  - only output redirection
  - both input and output redirection
- background and foreground modes

#162 - 02/13/2014 07:39 AM - Eugenie Lyzenko

- File alert-box.jpg added

> Eugenie: the modal message boxes are not supported by background mode. On linux pro -b -p with a MESSAGE ... VIEW-AS ALERT-BOX writes the data to standard output.

I think this is not correct. The example - batch_test4.p:

```
MESSAGE "Make a choice" VIEW-AS ALERT-BOX INFORMATION BUTTON YES-NO-CANCEL.
```

The result for both foreground and background modes in Linux. I have attached the screen-shot to prove. Any comments?

#163 - 02/13/2014 07:49 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Eugenie: the modal message boxes are not supported by background mode. On linux pro -b -p with a MESSAGE ... VIEW-AS ALERT-BOX writes the data to standard output.

> I think this is not correct. The example - batch_test4.p:
> [...]
> The result for both foreground and background modes in Linux. I have attached the screen-shot to prove. Any comments?

Your results are correct.

Sorry, for my case I forgot to mention that I was redirecting standard output to a file, using a pro -b -p batch_test4.p > batch.log command: in this case, the output is sent to the file. And this is the same as how appserver agents work: they write the standard output to a log file. That's why we

need to make sure all the possible combinations of background/foreground and input/output redirection are tested. Maybe there is a way of determining if standard output or input is redirected, from java/native code? To not add more configuration to the P2J client...

#164 - 02/13/2014 01:09 PM - Eugenie Lyzenko

- File evl_upd20140213a.zip added

The update uploaded that implements the messageBox() processing for background mode for you to review. The simulated reaction:

```
...
Linux
The foreground and background modes produces the error if output is not redirected
In redirected case the messageBox writes the message to the file.

Windows
The background and foreground modes in redirected case write output to the file
The foreground mode not redirected case generates the I/O error and
The background mode not redirected case write the message body to console
...
```

#165 - 02/14/2014 05:05 AM - Constantin Asofiei

Eugenie, I've used this configuration to start a client, on linux: client:driver:type=chui_batch client:mode:batch=true client:driver:background=true, and the client was started via java ... ClientDriver > client.log, to redirect the standard output to file. With your update, I still get the ** Attempt to write with no current output destination. (516) written to client.log - am I missing some configuration? Because I think there is something missing: we need to test if the standard output and input are redirected directly at the OS, not via the 4GL unnamed streams statements. You are using ThinClient.isRedirected(), which currently is not aware of OS-level redirection: this will not be seen by ThinClient.isRedirected and will not work for us.

And if I mentioned the unnamed streams, we need to add to the list of cases at notes 160/161 the cases when the OUTPUT TO and INPUT FROM are used to redirect the unnamed streams directly from 4GL.

#166 - 02/14/2014 06:04 AM - Eugenie Lyzenko

Constantin, The config options to start batch mode is:
swing="client:driver:type=chui_batch client:driver:background=true"
for background case
swing="client:driver:type=chui_batch"
for foreground case.

To test redirected mode please use batch_test4_1.p(or other tests with postfix "_1").

One note here more: when server is just started, the first client execution generates additional error message for ** Attempt to write with no current output destination. (516) in addition to correct messages. The second client execution doe not have this error. This is very strange behavior because in this case StopCondition is generated and server tries to reload application and then stops. Have you noted this on your system?

As to your note for external redirections like java ... ClientDriver > client.log - you are right, this case is missing form ThinClient point of view and considered as not redirected. Because for now java program does not know what exact is the output stream, console or file in this external case. I guess the additional config option is not a good idea, I'm thinking about something different(if this is possible at all).

#167 - 02/20/2014 06:15 AM - Constantin Asofiei

Eugenie, something else to check: a test like this works with no problem when invoked via an appserver or with batch programs when they have the standard output redirected:

```
def var i as int.
form i with frame f1.

on "1" of i in frame f1 do:
   message "trigger!".
end.

apply "1" to i in frame f1.
```

#168 - 02/20/2014 03:00 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> As to your note for external redirections like java ... ClientDriver > client.log - you are right, this case is missing form ThinClient point of view and considered as not redirected. Because for now java program does not know what exact is the output stream, console or file in this external case. I guess the additional config option is not a good idea, I'm thinking about something different(if this is possible at all).

I think this is becoming a show stopper, in terms of #2228 appserver usage. If you think is faster to just pass some client:driver:use_standard_output=true or client:driver:use_standard_input=true flags to the client (instead of determining this state from the native code), go ahead and use them. If these flags are set, just send all output to standard output and read the input from standard input.

#169 - 02/21/2014 11:03 AM - Eugenie Lyzenko

Looks like I have found the way to detect the externally redirected stdout. This is the native code. While the Windows has it's own way to do this:

```
...
   DWORD dwRes = 0;
   HANDLE hConsoleOut;

   hConsoleOut = GetStdHandle(STD_OUTPUT_HANDLE);
   dwRes = GetFileType(hConsoleOut);
   if ((dwRes & FILE_TYPE_CHAR) == FILE_TYPE_CHAR)
   {
      printf("The output device is console, type is: %d\n", dwRes);
   }
   else if ((dwRes & FILE_TYPE_DISK) == FILE_TYPE_DISK)
   {
      printf("The output device is disk file, type is: %d\n", dwRes);
   }
   else
```

```
   {
      printf("The output device type is: %d\n", dwRes);
   }
...
```

That works OK for both 32 and 64 bit Windows, there is a universal way to be used in Linux too:

```
...
   struct stat stRes;

   fstat(STDOUT_FILENO, &stRes);
   if (S_ISCHR(stRes.st_mode) != 0)
   {
      printf("The output device is console, stat is: %d\n", stRes.st_mode);
   }
   else if (S_ISBLK(stRes.st_mode) != 0)
   {
      printf("The output device is disk file, stat is: %d\n", stRes.st_mode);
   }
   else if (S_ISREG(stRes.st_mode) != 0)
   {
      printf("The output device is regular file, stat is: %d\n", stRes.st_mode);
   }
   else
   {
      printf("The output device is inknown, stat is: %d\n", stRes.st_mode);
   }
...
```

This approach gives S_ISCHR() != 0 for console and S_ISREG() != 0 for redirected console. Works in Linux and both Windows. So the condition

```
...
(S_ISCHR() == 0 && S_ISREG() != 0)
...
```

Gives redirected console by ">" command line option. And this means we are platform independent and do not need special code for Linux and Windows separately.

So the question is:
What will be the java/native places to implement this code. I would suggest to put java call into com/goldencode/util/PlatformHelper.java and native code into filesys.c(because we use file system tools to get info). Is it OK?

Let me know what do you think about this suggestion.

#170 - 02/21/2014 11:20 AM - Greg Shah

Good work!

> I would suggest to put java call into com/goldencode/util/PlatformHelper.java

Yes.

> and native code into filesys.c(because we use file system tools to get info).

Yes.

I like the approach.

#171 - 02/21/2014 03:52 PM - Eugenie Lyzenko

- File evl_upd20140221a.zip added

The update with suggested changes has been uploaded for review. Now I'm going to perform big testing for possible cases in background/foreground and different redirection types for Linux and Windows while you are reviewing. So this is possible not final version.

#172 - 02/21/2014 05:18 PM - Constantin Asofiei

Eugenie: we are using a spawn tool to launch the appserver agents, as they need to be automatically launched by the P2J server. The code is in native/spawn.c and the main process launching part was built by Marius on #1811 . See note 418 on #1811 regarding my concerns when redirecting the output for an appserver agent; please consult with Marius and determine if we need any changes in spawn.c, so that the child process has its output redirected, in case of appserver agents.

#173 - 02/24/2014 12:45 PM - Greg Shah

Code Review 0221a

I will let Constantin continue the discussion with you regarding the requirements for #2228.

My only concern with this update is that the approach in PlatformHelper will cause the native library to be loaded in the server, which is not something we are wanting to do.

Please move the call to native code out of the static initializer. Just use a Boolean instead of boolean for the consoleRedirected member. Default it to null. Then in isConsoleRedirected(), check for null and call the native code only if it is null. Of course, save the results to the consoleRedirected member and this will naturally avoid the future calls to the library. You should also synchronize this method.

#174 - 02/24/2014 02:17 PM - Eugenie Lyzenko

- File evl_upd20140224a.zip added

The new updated has been uploaded for you to check.

> My only concern with this update is that the approach in PlatformHelper will cause the native library to be loaded in the server, which is not something we are wanting to do.

I think only caller that use isConsoleRedirected() will cause the P2J library to be loaded. However may be PlatformHelper is not the best class to locate this native call. Because PlatformHelper is something p2j independent, right. May be better to use something from goldencode/p2j packages, OutputManager or other...

> Please move the call to native code out of the static initializer.

Yes, this was a bad idea. The PlatformHelper class initializes before P2J library loading. Rewritten to use Boolean cache value.

#175 - 02/24/2014 03:03 PM - Greg Shah

Code Review 0224a

1. The build.xml is missing from the update.

2. About this:

> I think only caller that use isConsoleRedirected() will cause the P2J library to be loaded.

In the new approach, that is true. But when the native code was in the static initializer, the first code that loaded the PlatformHelper class would cause this code to execute. The new design does not have that problem.

3. About this:

> However may be PlatformHelper is not the best class to locate this native call. Because PlatformHelper is something p2j independent, right.

Yes, I agree with this. Please move it to FileChecker.java. After moving it, the build.xml won't need changes.

4. I'll let Constantin review the ErrorWriterBatch.java changes.

#176 - 02/24/2014 06:12 PM - Eugenie Lyzenko

- File evl_upd20140224b.zip added

In this update the native call has been moved to FileChecker class. The build.xml does not need to be updated anymore.

#177 - 02/24/2014 06:17 PM - Greg Shah

Code Review 0224b

The code looks good. Make sure you have addressed everything needed by Constantin and he can review it in the morning. I'd like to get this into regression testing as soon as possible.

#178 - 02/24/2014 07:36 PM - Eugenie Lyzenko

- File evl_upd20140224c.zip added

This update is the result of additional debugging and testing in Windows. We have to handle(make a stub) setCursorStatus() in foreground batch mode to avoid JVM crash. And in this case if redirected - need to use println() for file logging.

#179 - 02/25/2014 08:03 AM - Constantin Asofiei

Eugenie, I'm OK with the changes in 0224c.zip. Together with the features which will be added by Marius on #1811 should get us past the STDOUT redirection in appserver mode.

Please summarize which statements you've tested (i.e. MESSAGE, DISPLAY, etc) and the difference in behaviour for the tested modes (i.e. STDOUT redirected, "pro -b", windows, linux, etc). Another feature (but I think we can postpone this for later) is the "STDIN redirected" mode, for batch processes. To think about it, I think 4GL hooks up the unnamed input and output streams to the redirected STDOUT and STDIN, when batch mode is used. But, if needed, we will give it another pass another time - for now, is enough if we can get past the appserver usages in #2228 (which your current update should do).

#180 - 02/25/2014 09:41 AM - Eugenie Lyzenko

The statements:
MESSAGE, DISPLAY, PUT SCREEN, SET, UPDATE, ENABLE all containing in batch test file set. On Windows/Linux background/foreground modes.

And I have found one more bug/mismatch. The DISPLAY statement put the data in console in Windows background batch mode and into file if redirected. In Linux redirected case causes writing data to the file(from DISPLAY statement), otherwise the error is generating.

Looking for a fix.

#181 - 02/25/2014 10:29 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> The statements:
> MESSAGE, DISPLAY, PUT SCREEN, SET, UPDATE, ENABLE all containing in batch test file set. On Windows/Linux background/foreground modes.

Have you tested the STDOUT redirection case in P2J? To start a batch process with STDOUT redirected, just add a > client.log at the end of the java command to start the P2J Client (which additionally is configured as a batch process).

#182 - 02/25/2014 10:34 AM - Eugenie Lyzenko

> Have you tested the STDOUT redirection case in P2J? To start a batch process with STDOUT redirected, just add a > client.log ...

Yes.

#183 - 02/25/2014 05:30 PM - Eugenie Lyzenko

- File evl_upd20140225a.zip added

The update for you review includes fix for DISPLAY statement handling. I've looked for different ways and the has one solution to process date in update screen stage. The other possible ways will require a lot of rework in very sensitive code in widget processing of redirected terminal usage. Yes, the fix is not a perfect and looks like a hack but other ways will take significant time to implement on my guess. On the other hand we concentrate all fixes for this issue related to batch processing in a batch related driver.

So if you do not mind I think the code is ready to testing for regressions. Let me know your conclusion.

#184 - 02/26/2014 01:30 AM - Constantin Asofiei

Eugenie, the changes in 0225a.zip are OK - go ahead with regression testing.

#185 - 02/26/2014 05:04 AM - Eugenie Lyzenko

The testing is in progress.

BTW, nave you noted the underline attribute for fill-in fields is not drawing. The example is the main MAJIC login screen for password. Can you check if your copy of the server/client has this issue or not? The harness can not detect this difference.

#186 - 02/26/2014 05:19 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> The testing is in progress.
>
> BTW, nave you noted the underline attribute for fill-in fields is not drawing. The example is the main MAJIC login screen for password. Can you check if your copy of the server/client has this issue or not? The harness can not detect this difference.

I'm not sure what problem you see: checking the MAJIC login screen from my instance on devsrv01 (with latest rev) and from your instance (I've started it from your account, with your P2J), the screens are the same. And the fill-in's have the underline....

#187 - 02/26/2014 08:39 AM - Eugenie Lyzenko

> I'm not sure what problem you see: checking the MAJIC login screen from my instance on devsrv01 (with latest rev) and from your instance (I've started it from your account, with your P2J), the screens are the same. And the fill-in's have the underline....

Sorry this was the wrong disturb as the result of my video driver configs. I confirm everything is OK in this area.

#188 - 02/26/2014 04:33 PM - Eugenie Lyzenko

The main part of testing has been passed without regressions. Running the CTRL-C part. Expect to finish today later.

#189 - 02/26/2014 06:22 PM - Eugenie Lyzenko

The CTRL-C part has been passed without regressions. The servers are now stopped.

#190 - 02/26/2014 06:33 PM - Greg Shah

Check in and distribute the change.

#191 - 02/26/2014 07:15 PM - Eugenie Lyzenko

Committed in bzr as 10482.

#192 - 11/16/2016 11:42 AM - Greg Shah

- Target version changed from Milestone 7 to Runtime Support for Server Features

Files

evl_upd20131105a.zip	19.4 KB	11/05/2013	Eugenie Lyzenko
evl_upd20131109a.zip	19.5 KB	11/10/2013	Eugenie Lyzenko
evl_upd20131119a.zip	49.3 KB	11/20/2013	Eugenie Lyzenko
evl_upd20131125a.zip	49.9 KB	11/25/2013	Eugenie Lyzenko
evl_upd20131127a.zip	162 KB	11/27/2013	Eugenie Lyzenko
evl_upd20131202a.zip	167 KB	12/02/2013	Eugenie Lyzenko
evl_upd20131205a.zip	238 KB	12/05/2013	Eugenie Lyzenko
evl_upd20131206a.zip	297 KB	12/07/2013	Eugenie Lyzenko
evl_upd20131208a.zip	297 KB	12/08/2013	Eugenie Lyzenko
evl_upd21031210a.zip	297 KB	12/11/2013	Eugenie Lyzenko
evl_upd20131213a.zip	307 KB	12/13/2013	Eugenie Lyzenko
evl_upd20131217a.zip	307 KB	12/18/2013	Eugenie Lyzenko
evl_upd20131219a.zip	308 KB	12/19/2013	Eugenie Lyzenko
evl_upd20131220a.zip	308 KB	12/20/2013	Eugenie Lyzenko
evl_upd20131220b.zip	308 KB	12/20/2013	Eugenie Lyzenko
evl_upd20131221a.zip	308 KB	12/21/2013	Eugenie Lyzenko
evl_upd20131221b.zip	308 KB	12/21/2013	Eugenie Lyzenko
evl_upd20140114a.zip	120 KB	01/14/2014	Eugenie Lyzenko
evl_upd20140115a.zip	258 KB	01/15/2014	Eugenie Lyzenko
evl_upd20140115b.zip	258 KB	01/15/2014	Eugenie Lyzenko
evl_upd20140116a.zip	273 KB	01/16/2014	Eugenie Lyzenko

evl_upd20140117a.zip	297 KB	01/18/2014	Eugenie Lyzenko
evl_upd20140120a.zip	299 KB	01/20/2014	Eugenie Lyzenko
evl_upd20140120b.zip	300 KB	01/21/2014	Eugenie Lyzenko
evl_upd20140121a.zip	300 KB	01/21/2014	Eugenie Lyzenko
evl_upd20140122a.zip	301 KB	01/22/2014	Eugenie Lyzenko
evl_upd20140124a.zip	300 KB	01/24/2014	Eugenie Lyzenko
testcases_20140210a.zip	9.88 KB	02/10/2014	Eugenie Lyzenko
batch_logs.zip	5.35 KB	02/11/2014	Constantin Asofiei
run_batch.p	1.61 KB	02/11/2014	Constantin Asofiei
ca_srv2.server.log	30.2 KB	02/11/2014	Constantin Asofiei
alert-box.jpg	319 KB	02/13/2014	Eugenie Lyzenko
evl_upd20140213a.zip	115 KB	02/13/2014	Eugenie Lyzenko
evl_upd20140221a.zip	125 KB	02/21/2014	Eugenie Lyzenko
evl_upd20140224a.zip	129 KB	02/24/2014	Eugenie Lyzenko
evl_upd20140224b.zip	131 KB	02/24/2014	Eugenie Lyzenko
evl_upd20140224c.zip	131 KB	02/25/2014	Eugenie Lyzenko
evl_upd20140225a.zip	127 KB	02/25/2014	Eugenie Lyzenko