Base Language - Feature #3467

implement clipboard stream support

02/06/2018 01:13 PM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Sergey Ivanovskiy	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to User Interface - Feature #1807: implement clipboard support	**Closed**	
Related to Base Language - Bug #3480: PUT STREAM statement uses SKIP modifier...	**New**	**02/21/2018**
Related to Base Language - Bug #3482: Pause message statement into hidden win...	**New**	**02/22/2018**

History

#1 - 02/06/2018 01:24 PM - Greg Shah

The 4GL provides the system CLIPBOARD as an output destination.

```
def var i as int.
def stream clippy.

output stream clippy to "clipboard".
do i = 1 to 5:
   put stream clippy "Line " i.
end.
output stream clippy close.
```

Right now, the stream implementation exists but it is stubbed out. See StreamFactory.openClipboardStream(). So, I don't think any conversion work is needed. But for the runtime, an actual backing stream needs to be created (which means new logic to create it in StreamDaemon) and the resulting output must be sent to the clipboard using our existing clipboard support.

You will have to experiment with the behavior of the 4GL, but my guess is that the copy to the clipboard only occurs when the stream is closed. Check this.

You should also test this with paged stream usage since that should be possible (set a page size in the OUTPUT TO).

#2 - 02/06/2018 01:25 PM - Greg Shah

- Related to Feature #1807: implement clipboard support added

#3 - 02/07/2018 06:17 AM - Sergey Ivanovskiy

Yes, I didn't observe the system clipboard changes by by pressing CTRL+V over the opened notepad if the write into the stream operation was paused and then proceeded.

```
def var i as int.
def stream clippy.

output stream clippy to "clipboard".
do i = 1 to 60:
   put stream clippy "Line " i "~r~n" SKIP.
end.

pause message "check the system clipboard buffer".
do i = 61 to 120:
   put stream clippy "Line " i "~r~n" SKIP.
end.

output stream clippy close.
```

#4 - 02/12/2018 08:56 AM - Sergey Ivanovskiy

Greg, can the clipboard stream be implemented such that it is the file stream opened for the temporary system file?

#5 - 02/13/2018 12:34 AM - Greg Shah

If necessary, yes.

But please first investigate the GUIPrinterStreamSupport implementation that eliminates the intermediate temporary file. Hynek can answer questions on this.

#6 - 02/13/2018 01:05 AM - Sergey Ivanovskiy

OK. It seems that temporary files are not required because ClipboardManager.setCliboardValue(String) is used to write data into the client's system clipboard. The target data will be stored in the string and then will be sent to the client.

#7 - 02/13/2018 05:47 AM - Sergey Ivanovskiy

A new unrelated issue that can be reproduced with the testcase from #3467-3 is that the default window isn't shown for this program and the pause statement doesn't produce its output into the window message area. The GUI 4GL opens the default window and prints its output into message area if this program is executed from the Windows command line terminal client (cmd.exe). In this case FWD doesn't show the default window for the tested Web and Swing clients. The Swing client is blocked, but the web client isn't blocked because it can get a pressed key from the JS client. The testcase is that

```
pause message "message".
```

#8 - 02/13/2018 07:19 AM - Greg Shah

Hynek: Didn't you make a change recently that is related to this?

#9 - 02/13/2018 10:38 AM - Hynek Cihlar

Greg Shah wrote:

> Hynek: Didn't you make a change recently that is related to this?

If I remember correctly I did some work on showing the default window when message statement was executed. But I don't recall the details. pause may be similar.

#10 - 02/13/2018 12:19 PM - Sergey Ivanovskiy

Greg, please review these changes rev. 11220 branch 3467a. I will work on #3467-7 in this branch if you have no objections.

#11 - 02/14/2018 12:56 AM - Greg Shah

> I will work on #3467-7 in this branch if you have no objections.

No objections.

Hynek: please monitor and review this aspect.

#12 - 02/14/2018 01:35 AM - Greg Shah

Code Review Task Branch 3467a Revision 11220

This is pretty good.

1. In ClipboardStream, why do writeByte(byte b), write(byte[] data) and write(byte[] data, int off, int len) all throw UnsupportedOperationException? This is something actively used by converted code. For example, the Stream.writeBlock() method is used with a memptr in code like EXPORT STREAM clippy my-memptr-var.. The bytes can be converted to characters using the JVM default character set. Later, we will have to add the other char conversion support but not for now.

2. Did you test paged clipboard streams as noted in #3467-1? If they work in the 4GL, then we would need to support that flag in the creation of the stream.

3. In ClipboardStream, there are several methods with throws UnsupportedOperationException, IOException (on a single line). Please split this over multiple lines as specified in the coding standards.

#13 - 02/14/2018 08:28 AM - Sergey Ivanovskiy

No, I missed the page size. I checked that with 4GL page-size for clipboard output stream inserts the form feed 0x0C byte after the specified number of lines (given by PAGE-SIZE)

```
output stream clippy to "clipboard" PAGE-SIZE 1 CONVERT SOURCE "ISO8859-1" TARGET "UTF-8".
```

Thank you for pointing on this issue. I didn't implement binary writing intentionally because it seems that the default encoding into string can have hidden issues. The following question is confusing me. Is it possible to alter the clipboard binary content by encoding it into string and then transmitting as 2-bytes character string to the remote client - JS client?

#14 - 02/14/2018 08:52 AM - Sergey Ivanovskiy

If PAGE-SIZE is 0 or isn't given, then the form feeds page breaks are not used. This observation contradicts to the langref.pdf that states

PAGE-SIZE{constant|VALUE (expression)}
Specifies the number of lines per page. The expression is a constant, field name, variable name, or expression whose value is an integer. The default number of lines per page is 56.
If you are using the TERMINAL option to direct output to the terminal, the default number of lines per page is the number of lines of TEXT widget. If you specify
PAGE-SIZE 0, the output is not paged in character mode; in a graphical interface, the default page size is used.s that fit in the window. If you specify a non-zero value for n , then the PAGED option is assumed.
If you specify PAGE-SIZE 0, the output is not paged in character mode; in a graphical interface, the default page size is used.

#15 - 02/14/2018 11:44 AM - Sergey Ivanovskiy

It seems that the PAGE-SIZE is used only if SKIP is used.

```
put stream clippy "Line " i "~r~n" SKIP.
```

The CR and LF doesn't produce this effect. Thus the data written by the following statement is not considered as a line without SKIP.

```
put stream clippy "Line " i "~r~n".
```

#16 - 02/14/2018 11:52 AM - Sergey Ivanovskiy

It seems that Stream doesn't have methods that can help to recognize line endings. From the previous note it follows that new line considered as a line in a page is not defined by the content at least for the clipboard stream. I have a problem here.

#17 - 02/14/2018 10:38 PM - Greg Shah

The Stream implementation already handles all of the form feed and newline behavior. Why do we need to do anything else?

Isn't the problem that the creation of the stream (in StreamFactory) does not honor the paged flag? If you enable that in the ClipboardStream, don't you get the correct result?

#18 - 02/15/2018 05:30 AM - Sergey Ivanovskiy

There are these methods in Stream related to paged output

```
  public void setPageSize(NumberType size);
  public void setPageSize(double size);
  public void setPageSize();
```

Greg, did you mean these methods to setup paged streams.
Another setBinary() method affects only readChar() according to java docs and public void setControl(boolean mode) can affect paged output.
I am confused by this implementation of FileStream.this.writeCh(char).

```
  public void writeCh(char ch)
   throws IOException
   {
      byte b = convert ? cc.toByte(ch) : (byte) ch;

      if (b == LF && !binary && EnvironmentOps.isUnderWindowsFamily())
      {
         // on Windows OS, non [binary] the newline char \n (LF, 0x0A) are encoded using 2 chars:
         // \r\n (CRLF, 0x0D 0x0A). On Linux and binary Windows, the newline character is not
         // altered
         write((byte) CR);
      }
      write(b);
   }
```

that alters output in the case of Windows OS and non binary stream. Is the output stream without explicit BINARY supposed to have binary=false?

#19 - 02/15/2018 05:42 AM - Sergey Ivanovskiy

Greg Shah wrote:

> > I will work on #3467-7 in this branch if you have no objections.
>
> No objections.
>
> Hynek: please monitor and review this aspect.

This code fragment of Window.this.message(...)

```
   public MessageReturnValue message(String       text,
                                     boolean      set,
                                     BaseDataType var,
                                     boolean      auto,
                                     String       format,
                                     Color        cs)
   {
      ThinClient tc = ThinClient.getInstance();

      // only show the default window
      if (!isVisible() && getId().equals(WidgetId.DEFAULT_WINDOW_ID))
      {
         // query redirection
         boolean switched = tc.isRedirected();
         int redir = -1;

         // in redirected mode we output to both the terminal and to the stream, but we output
         // the unmodified text, not the text that may have been truncated above
         if (switched)
         {
            redir = tc.switchOutput(-1, false);
         }

         ThinClient.getInstance().independentEventDrawingBracket(this, new Runnable()
         {
            public void run()
            {
               show();
            }
         });

         if (switched)
         {
            tc.switchOutput(redir, false);
         }
      }
............................
}
```

works for this test case

```
message "The default window".
```

I used this part for the pause test case

```
pause message "Wait until key is pressed".
```

Please review commited rev 11222. I need your help to understand if the code above can be generalized for pause. There are operation braces tc.switchOutput(-1, false) and tc.switchOutput(redir, false); that are not used in ThinClient.this.pause(int seconds, String text, boolean noMessage, TopLevelWindow wnd) implementation now.

#20 - 02/15/2018 08:37 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> I need your help to understand if the code above can be generalized for pause. There are operation braces tc.switchOutput(-1, false) and tc.switchOutput(redir, false); that are not used in ThinClient.this.pause(int seconds, String text, boolean noMessage, TopLevelWindow wnd) implementation now.

The switchOutput calls are there to restore redirected output so that the message appears in the window. Double check how PAUSE should behave in this regard. Before the PAUSE statement redirect output to a file. Check whether the message appears in the redirected output, whether the output is shown in the window (default or a dynamically created window) and whether the invisible (default or dynamically created) window will be shown.

#21 - 02/15/2018 08:53 AM - Hynek Cihlar

Hynek Cihlar wrote:

> The switchOutput calls are there to restore redirected output so that the message appears in the window. Double check how PAUSE should behave in this regard. Before the PAUSE statement redirect output to a file. Check whether the message appears in the redirected output, whether the output is shown in the window (default or a dynamically created window) and whether the invisible (default or dynamically created) window will be shown.

Sergey, when you are at it. Please look at the following test case:

```
def var h as handle.
create window h.
pause message "test" in window h.
message "another statement".
```

In native Progress the application doesn't get paused at the PAUSE statement and continues with the next statement, or exits if there is no further statement to execute.

In FWD however, the client seems to be paused.

If you don't find the solution quickly, please do not bother further and create a new issue for this.

#22 - 02/19/2018 07:30 AM - Greg Shah

Are there remaining issues with PAUSE here? If you have found any other problems with PAUSE, please create a new issue.

Hynek: Are you OK with the TC change in 3467a rev 11225?

> Greg, did you mean these methods to setup paged streams.

Yes, but I meant not only that. If a stream is in paged mode (PAGED is set in the 4GL and/or PAGE-SIZE is set in the 4GL), the output methods honor this by automatically outputting a form feed when the number of lines exceeds the page size limit.

The specific stream implementation (printer, clipboard, file, process...) does not have to do anything to get this behavior.

Now that you have enabled the paged flag and the explicit page size to the StreamFactory.openClipboardStream() methods, does the paged output now work with clipboard?

> Is the output stream without explicit BINARY supposed to have binary=false?

Yes. 4GL streams are text mode by default.

#23 - 02/19/2018 08:44 AM - Hynek Cihlar

Greg Shah wrote:

> Are there remaining issues with PAUSE here? If you have found any other problems with PAUSE, please create a new issue.
>
> Hynek: Are you OK with the TC change in 3467a rev 11225?

I did a review of the changes, but there are unresolved functional concerns in note 19 and 20.

#24 - 02/19/2018 09:39 AM - Sergey Ivanovskiy

- File clipboard_stream.p added

> Now that you have enabled the paged flag and the explicit page size to the StreamFactory.openClipboardStream() methods, does the paged output now work with clipboard?

Greg, please review this rev 11226. I compared how the 4GL and FWD can process this program clipboard_stream.p. 4GL counts lines using SKIP statements. FWD alters windows line endings "~r~n" to be "~n"(0x0a) and doesn't write 0x0c page breaks for this example. The number of pages is number of skipped lines (number lines with SKIP) divided by 3. It seems that Stream has issues or my ClipboardStream is incorrect. I followed your guide answers and tested with help of bless - bless - graphical hexadecimal Gtk# editor.

#25 - 02/19/2018 09:42 AM - Sergey Ivanovskiy

The note #3467-24 was updated.

#26 - 02/19/2018 09:58 AM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> I did a review of the changes, but there are unresolved functional concerns in note 19 and 20.

I didn't find in 4GL documentations how to redirect PAUSE MESSAGE into STREAM. It seems that these modifiers STREAM-IO and SCREEN-IO can't be applied in this case. Finally I don't know how to redirect into file stream the pause message.

#27 - 02/19/2018 10:33 AM - Hynek Cihlar

Sergey Ivanovskiy wrote:

> Hynek Cihlar wrote:
>
> > I did a review of the changes, but there are unresolved functional concerns in note 19 and 20.
>
> I didn't find in 4GL documentations how to redirect PAUSE MESSAGE into STREAM. It seems that these modifiers STREAM-IO and SCREEN-IO can't be applied in this case. Finally I don't know how to redirect into file stream the pause message.

I don't think you can redirect a pause message. Still we want to test whether the code changes don't break the redirected mode when pause is issued.

Just do a few variations of the following test, with a default window, a dynamically created visible or non-visible window.

```
output to "out.txt".
pause message "test".
```

#28 - 02/19/2018 03:42 PM - Greg Shah

> It seems that Stream has issues or my ClipboardStream is incorrect.

Please post the output from FWD and from the 4GL as txt files.

In addition, change the code to use this:

```
output stream clippy to my-test-file.txt PAGED PAGE-SIZE 3 CONVERT SOURCE "ISO8859-1" TARGET "UTF-8".
```

And then retest. Upload the results here as txt files. If output to file shows the same behavior, then the issue is in our stream support and the clipboard is fine.

#29 - 02/19/2018 04:21 PM - Sergey Ivanovskiy

- File 4gl_to_file_output.txt added

- File clipboard_stream_fwd_output.txt added

- File clipboard_stream_4gl_output.txt added

- File fwd_to_file_output.txt added

Please look at these test results:
1) Clipboard
clipboard_stream_fwd_output.txt is a clipboard stream output produced by FWD
clipboard_stream_4gl_output.txt is a clipboard stream output produced by 4GL
2) Files
fwd_to_file_output.txt is a file stream output produced by FWD
4gl_to_file_output.txt is a file stream output produced by 4GL

#30 - 02/19/2018 07:15 PM - Greg Shah

These results show a general failure in our line/form feeds. It is not specific to clipboard (although the 4GL does have some difference in behavior for the clipboard).

Please do the following:

- check in both example programs (one for files and one for clipboard)
- open a new task describe the failure, make sure to upload these results (all 4 files) to show the problem
- mark it related to this task and make me a watcher

We are not going to work this now.

#31 - 02/21/2018 10:19 AM - Sergey Ivanovskiy

- Related to Bug #3480: PUT STREAM statement uses SKIP modifier to complete a line output added

#32 - 02/22/2018 10:59 AM - Sergey Ivanovskiy

- Related to Bug #3482: Pause message statement into hidden window blocks the program flow. added

#33 - 02/22/2018 11:09 AM - Sergey Ivanovskiy

Hynek Cihlar wrote:

> Sergey Ivanovskiy wrote:
>
>> Hynek Cihlar wrote:
>>
>>> I did a review of the changes, but there are unresolved functional concerns in note 19 and 20.
>>
>> I didn't find in 4GL documentations how to redirect PAUSE MESSAGE into STREAM. It seems that these modifiers STREAM-IO and SCREEN-IO can't be applied in this case. Finally I don't know how to redirect into file stream the pause message.
>
> I don't think you can redirect a pause message. Still we want to test whether the code changes don't break the redirected mode when pause is issued.
>
> Just do a few variations of the following test, with a default window, a dynamically created visible or non-visible window.

I tested these cases and opened new issue #3482. It seems that #3467 is complete except these new issues #3480 and #3482.

#34 - 02/22/2018 11:48 AM - Greg Shah

Hynek Cihlar wrote:

> Greg Shah wrote:
>
>> Are there remaining issues with PAUSE here? If you have found any other problems with PAUSE, please create a new issue.
>>
>> Hynek: Are you OK with the TC change in 3467a rev 11225?

> I did a review of the changes, but there are unresolved functional concerns in note 19 and 20.

Are the concerns addressed by #3482 or are there different concerns?

In other words, is it OK to leave the rev 11225 changes in this branch?

#35 - 02/22/2018 11:48 AM - Greg Shah

Code Review Task Branch 3467a Revision 11226

I'm OK with the changes.

#36 - 02/22/2018 01:11 PM - Hynek Cihlar

Greg Shah wrote:

> Are the concerns addressed by #3482 or are there different concerns?
>
> In other words, is it OK to leave the rev 11225 changes in this branch?

Yes, I think the changes can stay, no more concerns here.

#37 - 02/22/2018 01:22 PM - Greg Shah

Sergey: Please add a history entry to ThinClient. Then please start ChUI runtime regression testing. I don't expect issues, but it is good to be safe.

#38 - 02/23/2018 12:13 AM - Sergey Ivanovskiy

I did the fist try of run-time regression tests. All parts were not passed: all ctrl_c part was failed after ~ 2 hours and main part was failed after 18 minutes of test execution. Planning to the next try.

#39 - 02/23/2018 07:48 AM - Greg Shah

Don't worry about the ctrl-c tests. Just run the main regression tests.

#40 - 02/23/2018 08:02 AM - Sergey Ivanovskiy

The second try of the main regression tests produced results with 6 failed tests (results/20180223_021610.zip). Planning to start testing again.

#41 - 02/23/2018 02:59 PM - Sergey Ivanovskiy

The third try had this non-testing error

```
main-regression:
     [exec] ** Clearing sbi folder...
     [exec] rm: cannot remove '/home/sbi/web*.dat': No such file or directory
     [exec] ** Running main part...
     [exec]
     [exec] Running test plan... ESC M : R is 7, tm is 7, bm is 19
     [exec] ESC M : R is 7, tm is 7, bm is 19
     [exec] ESC M : R is 0, tm is 0, bm is 3
     [exec] ESC M : R is 1, tm is 1, bm is 19
     [exec] ESC M : R is 1, tm is 1, bm is 19
     [exec] ESC M : R is 1, tm is 1, bm is 19
     [exec] ESC M : R is 4, tm is 4, bm is 19
     [exec] ESC M : R is 13, tm is 13, bm is 20
     [exec] ESC M : R is 7, tm is 7, bm is 20
     [exec] Non-testing errors (CONNECT_FAILURE, complete == true) occured during test execution:
     [exec] com.jcraft.jsch.JSchException: channel is not opened.
     [exec]         at com.jcraft.jsch.Channel.sendChannelOpen(Channel.java:765)
     [exec]         at com.jcraft.jsch.Channel.connect(Channel.java:151)
     [exec]         at com.goldencode.harness.transport.SSH2Transport.connect(SSH2Transport.java:79)
     [exec]         at com.goldencode.harness.transport.TransportManager.connect(TransportManager.java:69)
     [exec]         at com.goldencode.harness.Driver.<init>(Driver.java:83)
     [exec]         at com.goldencode.harness.TestSet.run(TestSet.java:257)
     [exec]         at java.lang.Thread.run(Thread.java:748)
     [exec]
     [exec] ** MAIN part has failed!
```

#42 - 02/28/2018 09:16 AM - Sergey Ivanovskiy

The task branch 3467a rev 11233 passed main regression tests as a result of several runs and merged into the trunc as rev. 11229.

#43 - 02/28/2018 09:21 AM - Greg Shah

- Status changed from New to Closed

- % Done changed from 0 to 100

Files

clipboard_stream.p	491 Bytes	02/19/2018	Sergey Ivanovskiy
clipboard_stream_fwd_output.txt	3.44 KB	02/19/2018	Sergey Ivanovskiy
clipboard_stream_4gl_output.txt	3.73 KB	02/19/2018	Sergey Ivanovskiy
fwd_to_file_output.txt	3.65 KB	02/19/2018	Sergey Ivanovskiy
4gl_to_file_output.txt	4 KB	02/19/2018	Sergey Ivanovskiy