Base Language - Bug #4628

redirection doesn't seem to close an output stream on a block retry

04/24/2020 03:57 PM - Roger Borrello

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Roger Borrello	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

Related issues:

Related to Base Language - Feature #4395: add support for KEEP-MESSAGES option	**Closed**

History

#1 - 04/24/2020 03:57 PM - Roger Borrello

- Related to Feature #4395: add support for KEEP-MESSAGES option added

#2 - 04/24/2020 04:02 PM - Roger Borrello

One of the behaviors of the 4GL and FWD when the error condition raised from i = integer("garbage"). is to jump execution from the beginning, and the statements after that are not executed. When I added no-error, it created a situation where no errors were logged in the redirection. All good so far, but in FWD, one of the lines that isn't processed after the ** Invalid character in numeric input g. (76) is thrown is the output close. Because of this, the stream is still maintaining the KEEP-MESSAGES option, and when the program re-executes the message "Redirected?" set redir as logical. the text is suppressed from being displayed.

It's difficult to determine how the 4GL is actually handling this. Perhaps it implicitly closes the stream, and we don't?

Testcase: ./uast/io/advanced_redirected_output.p

#3 - 04/24/2020 06:13 PM - Greg Shah

Please cut this down to be something much smaller. I seriously doubt that more than 10% is needed. No prompting of the user is needed. Try with just something as simple as this:

```
def var i as int.
output to whatevs.txt.
i = integer("garbage").
output close.
```

To make it easier to fix the problem we pare away all the unnecessary junk.

#4 - 04/26/2020 08:31 AM - Greg Shah

- Subject changed from Redirection doesn't seem to close an output stream on a block re-execution to redirection doesn't seem to close an output stream on a block retry

RETRY is a very specific thing. Better to say that than to say re-execution, which has no 4GL meaning. The problem is likely to be in how Stream implements Finalizable (Stream.retry() is empty).

#5 - 04/27/2020 11:36 AM - Roger Borrello

Greg Shah wrote:

> Please cut this down to be something much smaller. I seriously doubt that more than 10% is needed. No prompting of the user is needed. Try with just something as simple as this:
>
> [...]
>
> To make it easier to fix the problem we pare away all the unnecessary junk.

The cut-down testcase does not behave in the same manner. The application exits (as does the 4GL). There must be something around a retry-block that isn't in the simplified case. I'm cutting down the more elaborate testcase until the behavior changes.

#6 - 04/27/2020 12:21 PM - Roger Borrello

New testcase: ./uast/io/redirected_stream_close_on_retry.p:

```
def var i as int.
message "Keep-Messages?" set bogus as logical.
output to ./test.txt.
output close.
output to ./test.txt keep-messages append.
i = integer("garbage").
output close.
message "Done.".
```

On 4GL, output (with 2 retries):

```
** Invalid character in numeric input g. (76)
Keep-Messages?     no
** ./test.txt already has a conflicting use. (99)
Keep-Messages?     no
** ./test.txt already has a conflicting use. (99)
```

On FWD:

```
rfb@rfb:~/testcases/deploy/client$ cat test.txt
```

```
** Invalid character in numeric input g. (76)
** Invalid character in numeric input g. (76)
```

The error message already has a conflicting use. (99) is not being reported by FWD, which must be an unsupported error condition. Would that point to 4GL noticing the stream is already open, logging the message "Keep-Messages?" message and response to it, logging that fact the stream is already opened, and proceeding on, including retries ad-nauseum?

Also, the error message is not re-logged. It isn't logged to the screen either. It just goes down the bit-bucket.

#7 - 04/27/2020 04:20 PM - Greg Shah

> The error message already has a conflicting use. (99) is not being reported by FWD, which must be an unsupported error condition.

Probably. Trying to re-open a destination that is already open seems to cause this.

> Would that point to 4GL noticing the stream is already open, logging the message "Keep-Messages?" message and response to it, logging that fact the stream is already opened, and proceeding on,

Yes.

> including retries ad-nauseum?

Will either 4GL or FWD continue retrying indefinitely?

> Also, the error message is not re-logged. It isn't logged to the screen either. It just goes down the bit-bucket.

I doubt it. In the 4GL, the error 76 only does happen once. After that in the 4GL the error 99 causes the retry and the code never executes the i = integer("garbage")..

This example does show that we are missing the raising of an error. It also shows that we are not writing out the message content to the already re-directed stream (this may be a known problem).

But it does not show the retry behavior you were describing.

#8 - 04/27/2020 05:16 PM - Constantin Asofiei

Roger, anything related to retry should be ran from the command line and not the procedure editor.

If you run it from the procedure editor, as the procedure editor is a 4GL program itself, you are not in a 'clean environment' to test for RETRY.

#9 - 04/29/2020 04:46 PM - Roger Borrello

- Assignee set to Roger Borrello

#10 - 04/29/2020 05:31 PM - Roger Borrello

Constantin Asofiei wrote:

> Roger, anything related to retry should be ran from the command line and not the procedure editor.
>
> If you run it from the procedure editor, as the procedure editor is a 4GL program itself, you are not in a 'clean environment' to test for RETRY.

Thanks for that info. I ran using prowin.exe -p. Is that the correct method? It shows in the output file the error message only once, then retries, and re-opens an existing stream, logging the *conflicting use* message. When I removed keep-messages the redirection behavior was the same, but the error messages showed up in the message area.

```
Z:\testcases\uast\io>C:\Progress\OE116_64\bin\prowin.exe -p redirected_stream_close_on_retry.p
Z:\testcases\uast\io>type test.txt
** Invalid character in numeric input g. (76)
Yes or no?     no
** ./test.txt already has a conflicting use. (99)
Yes or no?     no
** ./test.txt already has a conflicting use. (99)
Yes or no?     yes
** ./test.txt already has a conflicting use. (99)
```

Testcase:

```
def var i as int.
message "Yes or no?" set bogus as logical.
output to ./test.txt keep-messages append.
i = integer("garbage").
output close.
message "Done.".
```

FWD output:

```
** Invalid character in numeric input g. (76)
** Invalid character in numeric input g. (76)
** Invalid character in numeric input g. (76)
** Invalid character in numeric input g. (76)
```

When walking through the server Conversation block, it looks like when we get to the retry, we get a brand new stream from the client. The first time through, the id was 0, the second time it is 1. Will I have to debug the client?

#11 - 04/30/2020 10:44 AM - Greg Shah

As noted above, the fact that the MESSAGE SET output is not redirected in FWD is a known issue. We've never fixed it because it is only something we see in silly testcases. Real customer code has not exposed it, presumeably because it is not very useful for real life business logic.

> When walking through the server Conversation block, it looks like when we get to the retry, we get a brand new stream from the client. The first time through, the id was 0, the second time it is 1. Will I have to debug the client?

There is a real problem here, but it has nothing to do with RETRY. We should be generating an error when trying to open the same file twice. The 4GL in fact DOES NOT close the stream on RETRY. Otherwise there would be no ** ./test.txt already has a conflicting use. (99). I do think we should add that error to our implementation.

You still need to find a way to prove that the 4GL "automatically closes a stream on RETRY". If that is a real problem I want to fix it too.

#12 - 05/01/2020 02:14 PM - Roger Borrello

Just documenting some articles related to this:

- This article (https://knowledgebase.progress.com/articles/Article/P23598) indicates: *Note than when running procedure terminates, the output file is closed whether or not an explicit OUTPUT CLOSE statement is included.*
- This one is a ridiculously simple example: https://www.progresstalk.com/threads/already-has-a-conflicting-use-99.115004/
- This seems to be fixed, because it doesn't behave as notes, it behaves as you'd expect: https://knowledgebase.progress.com/articles/Article/000055787

#13 - 01/18/2022 09:14 AM - Roger Borrello

Just a note to indicate the fixes for #5880 do not appear to affect this bug at all.