Base Language - Feature #1630

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

implement support for UNBUFFERED in I/O statements

10/21/2012 10:36 AM - Greg Shah

Status:	Closed	**Start date:**	01/29/2014
Priority:	Normal	**Due date:**	02/07/2014
Assignee:	Evgeny Kiselev	**% Done:**	100%
Category:		**Estimated time:**	32.00 hours
Target version:	Cleanup and Stablization for Server Features		
billable:	No	**vendor_id:**	GCD

Description

History

#1 - 10/21/2012 10:37 AM - Greg Shah

- Estimated time changed from 12.00 to 16.00

#2 - 10/31/2012 01:41 PM - Greg Shah

- Target version set to Milestone 7

#3 - 01/14/2013 06:30 PM - Greg Shah

Conversion is already supported. But the runtime probably needs changes to support this option. Testcases will have to be written to explore what this does.

#4 - 04/25/2013 11:16 AM - Greg Shah

- Target version changed from Milestone 7 to Milestone 11

#5 - 05/02/2013 07:08 PM - Greg Shah

- Due date set to 09/05/2013

- Estimated time changed from 16.00 to 32.00

- Parent task set to #1623

- Assignee set to Evgeny Kiselev

- Start date set to 08/27/2013

Only 2 forms are in use in the server project:

OUTPUT TO filename APPEND UNBUFFERED.
OUTPUT TO filename UNBUFFERED.

Sometimes filename is hard coded and sometimes it is specified as VALUE. But that should not affect the UNBUFFERED case. I suspect the APPEND doesn't matter either, but testcases will be needed to prove it.

#6 - 01/21/2014 11:30 AM - Greg Shah

- Due date changed from 09/05/2013 to 02/07/2014

- Start date changed from 08/27/2013 to 01/29/2014

#7 - 05/09/2014 06:42 PM - Evgeny Kiselev

Do I need to test only output or all places there can be used unbuffered ? (like input, input-output and etc)

#8 - 05/12/2014 02:16 PM - Greg Shah

Check all uses (not just OUTPUT). I would hope that it just means that we flush with every write (synchronously).

#9 - 05/19/2014 08:50 PM - Evgeny Kiselev

- File unbuffered_testcases20140519a.zip added

UNBUFFERED statement is not working properly in p2j. This keyword is properly converted, but worked different.
One example from update(other tests are not working too):

```
function checkFile returns int (fileName as character, c as character):
   define variable s as memptr no-undo.
   copy-lob from file fileName to s no-convert.
   /* on this line there is no data in file(file is created), but in 4gl all need data is already in file*/
   if (c ne string(s)) then
   do:
      message "ERROR: " string(s) .
      return -1.
   end.

   return 1.
end.

def var res as int.
def var s1 as character no-undo.
def var s2 as character no-undo.

s1 = "10".
s2 = "SoMeTeSt".

define stream rep.
output stream rep to unbuff_file.txt unbuffered.

put stream rep unformatted s1.

res = checkFile("unbuff_file.txt", s1).

if (res eq -1) then message "error1".

put stream rep unformatted s1.
res = checkFile("unbuff_file.txt", s1 + s1).

if (res eq -1) then message "error1".

put stream rep unformatted s2.
res = checkFile("unbuff_file.txt", s1 + s1 + s2).

if (res eq -1) then message "error1".
```

Also I've not tested input stream with unbuffered property. Do you have idea how I can test it ? I can write some program for example and try to read something.

UNBUFFERED property is allowed in:
INPUT FROM statement (not tested yet)
INPUT THROUGH statement (not tested yet)
INPUT-OUTPUT THROUGH statement (tested)
OUTPUT THROUGH statement (tested)
OUTPUT TO statement (tested)

#10 - 05/20/2014 10:36 AM - Greg Shah

It is expected that P2J is not working. The runtime support for UNBUFFERED is not yet implemented. Your testcases looks good for the I/O and O cases, except you need to make sure that OUTPUT THROUGH is included (I don't see it in the testcases).

Also, please add a test for OUTPUT TO "somefile.txt" UNBUFFERED (redirects the unnamed stream from the terminal to a file) and then use DISPLAY to generate formatted output to that file (DISPLAY uses a frame to display "report-like" output in this case).

To implement UNBUFFERED, I think that you need to identify the possible places in Stream.java (and possibly in the subclasses) where output occurs and ensure that we flush immediately using the underlying Java stream facilities. There is some trickiness in the redirected terminal case because we hide the buffering in the P2J code, but I think it might be as simple as calling sync() after every output operation.

You can look at BufferSizeManager, but this doesn't do real buffering. It just tracks buffer related events and allows ProcessStream instances to detect a specific case when they have to generate a deferred STOP condition. You should not have to do anything that class.

> Also I've not tested input stream with unbuffered property. Do you have idea how I can test it ? I can write some program for example and try to read something.

It is a good question. Yes, please do find a way to test it. Sometimes the 4GL has some "nonsense" support that doesn't really have meaning. In this case, it is hard to see what UNBUFFERED input might mean. From the 4GL reference for INPUT FROM/INPUT THROUGH (the OS-DIR portion is only for INPUT FROM):

```
Reads one character at a time from a normally buffered data source, such as a
file. Use the UNBUFFERED option only when you can intermingle the input
operations of a UNIX process, invoked with the ABL UNIX statement, with the
input that follows the ABL INPUT FROM statement.
...
When using the OS-DIR option, the UNBUFFERED option is ignored. OS-DIR
always buffers exactly one filename at a time.
```

Language statements that can be used to read from streams:

- READ-KEY (one char at a time)
- IMPORT (can read either one line at a time in UNFORMATTED mode or as a list of formatted values that are read into vars/db fields)
- PROMPT-FOR/SET/UPDATE (reads formatted values into widgets in a frame; the widgets are backed by vars or db fields)

Perhaps using UNBUFFERED will change how these return (immediate returns instead of blocking) or perhaps it will allow partial reads. You will have to experiment to see if you can find a difference. As with NO-ECHO, you will probably need to run the same test with and without UNBUFFERED (using a logical flag to control the choice at runtime).

#11 - 05/25/2014 08:50 PM - Evgeny Kiselev

- File evk_upd20140523a.zip added

> Also, please add a test for OUTPUT TO "somefile.txt" UNBUFFERED (redirects the unnamed stream from the terminal to a file) and then use DISPLAY to generate formatted output to that file (DISPLAY uses a frame to display "report-like" output in this case).

Tested, works fine and same.

> To implement UNBUFFERED, I think that you need to identify the possible places in Stream.java (and possibly in the subclasses) where output occurs and ensure that we flush immediately using the underlying Java stream facilities. There is some trickiness in the redirected terminal case because we hide the buffering in the P2J code, but I think it might be as simple as calling sync() after every output operation.

I've implemented UNBUFFERED option in each low level stream class. Because some of them has each buffer like FileStream. ProcessStream has already flushed data after each write(he used OutWriter and OutStream). I've tested it and it's work properly. TerminalStream has also flushed (sync()) data after each write.

With input operation it was a little bit tricky. There is no any special method flush() for InputStream(I mean java) we can only protect us from buffereing if we will use java unbuffered Readers like FileInputStream, InputStreamReader, FileReader, I've rechecked and retested all input operations and we didn't use any buffered streams.

> Perhaps using UNBUFFERED will change how these return (immediate returns instead of blocking) or perhaps it will allow partial reads. You will have to experiment to see if you can find a difference. As with NO-ECHO, you will probably need to run the same test with and without UNBUFFERED (using a logical flag to control the choice at runtime).

Tested 50%. Don't see any difference atm.

#12 - 05/28/2014 01:45 PM - Greg Shah

Code Review 0523a

The changes look good. I have some minor comments and questions:

1. Please merge the code up to the latest P2J level in bzr (Stream.java is missing changes).

2. In both classes, please place protected methods after public methods and before private methods, as our coding standards dictate.

3. If I understood your comments correctly, the ProcessStream and TerminalStream require no changes because in our P2J implementation they are already always unbuffered. Is that right?

4. What about PrinterStream?

#13 - 05/29/2014 08:10 PM - Evgeny Kiselev

- File evk_upd20140528a.zip added

1. Please merge the code up to the latest P2J level in bzr (Stream.java is missing changes).
2. In both classes, please place protected methods after public methods and before private methods, as our coding standards dictate.

done

> 3. If I understood your comments correctly, the ProcessStream and TerminalStream require no changes because in our P2J implementation they are already always unbuffered. Is that right?

Yes, in this classes unbuffered operation is already implemented:
class ProcessStream:
Used IOHelper interface to write or read something. Subclass OutWriter and OutStream call flush() after every write operation. Also InStream and InReader cannot write at all.

class TerminalStream:
This class always call sync.refresh() after each write. There is no flush like in common IO operation, but this method do the same.

> 4. What about PrinterStream?

I've missed PrinterStream because this class is not Subclass of Stream class. But after looking into code, I've found that this class using ProcessStream class and wrap all other streams into subclasses of IOHelper (which are always using flash())

#14 - 05/30/2014 08:11 AM - Greg Shah

Code Review 0528a

The code looks good.

Have you finished testing all of your testcases? If it is fully working in all of your testcases, please start runtime regression testing. Conversion testing is not necessary.

#15 - 05/30/2014 09:09 PM - Evgeny Kiselev

Greg Shah wrote:

> Code Review 0528a
>
> The code looks good.
>
> Have you finished testing all of your testcases? If it is fully working in all of your testcases, please start runtime regression testing. Conversion testing is not necessary.

Yes, I've finished testing and now running regression testing.

#16 - 06/01/2014 08:08 PM - Evgeny Kiselev

Update evk_upd20140528a.zip has passed regression testing and ready to commit.

#17 - 06/02/2014 06:34 AM - Greg Shah

Great! Please do commit it and distribute it.

#18 - 06/02/2014 08:12 PM - Evgeny Kiselev

Committed to bzr rev. 10542.

#19 - 06/03/2014 05:04 AM - Greg Shah

- % Done changed from 0 to 100

- Status changed from New to Closed

#20 - 11/16/2016 12:07 PM - Greg Shah

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

Files

unbuffered_testcases20140519a.zip	2.23 KB	05/20/2014	Evgeny Kiselev
evk_upd20140523a.zip	41.6 KB	05/26/2014	Evgeny Kiselev
evk_upd20140528a.zip	41.7 KB	05/30/2014	Evgeny Kiselev