Bugs - Bug #1442

stream re-opened to the same file flushes previous buffer

05/01/2012 08:03 AM - Constantin Asofiei

Status:	WIP	**Start date:**	04/23/2012
Priority:	Normal	**Due date:**	
Assignee:	Constantin Asofiei	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 05/01/2012 08:03 AM - Constantin Asofiei

Refs #1439

I've found a case in P2J which I'm not sure if it's a Progress "feature" or a P2J bug. Assume the following scenario:
1: output stream s to "a.txt".
2: // write 5 lines to stream S
3: output stream s to "a.txt". /* note that the same S var is used and stream S is not previously closed */
4: // write 2 lines to stream S
5: output stream s close.

Problem is, at line 3, P2J does the following:
a. instantiates another internal stream to output to file F
b. as the file is open in non-append mode, it clears it
c. when assigning the new internal stream reference, the stream variable notices that its previous internal stream was not closed, so it closes it (thus flushing any remaining data to the file)
d. sets the internal stream reference to the stream created at a)

The output of such a case is:
2
2
5
5
5

where "2" is from the "write 2 lines" code and "5" is from the "write 5 lines" code. In my opinion, the output should have been:
2
2

i.e. the previous output from "write 5 lines" should have been discarded when the stream is re-assigned (even to the same file).

If this is a P2J bug, problem is the stream doesn't close its previous internal stream before the new internal stream is created.

#3

#2 - 10/23/2012 02:28 PM - Greg Shah

- File ca_upd20120501a.zip added

-------- Original Message --------
Subject: proposed update [ca_upd20120501a.zip]
Date: Tue, 01 May 2012 16:01:32 +0300
From: Constantin Asofiei <ca@goldencode.com>
Reply-To: ca@goldencode.com
To: Greg Shah <ges@goldencode.com>, "Faulhaber, Eric" <ecf@goldencode.com>

Greg,

Attached update is for the stream issue. I'll let you know the harness
results after they are done.

Thanks,
Constantin

#3 - 10/23/2012 02:31 PM - Greg Shah

- File ca_upd20120719a.zip added

-------- Original Message --------
Subject: Re: proposed update [ca_upd20120719a.zip]
Date: Thu, 19 Jul 2012 17:31:27 +0300
From: Constantin Asofiei <ca@goldencode.com>
Reply-To: ca@goldencode.com
To: Greg Shah <ges@goldencode.com>
CC: Faulhaber, Eric <ecf@goldencode.com>

Greg,

Please discard the ca_upd20120501a.zip update and use this one (some
javadocs fixes I missed and a cleaner version of the
UnnamedStreams.openFileIn/Out APIs). Documentation is built on this version.

Thanks,
Constantin

#4 - 10/23/2012 02:31 PM - Greg Shah

Change summary:

-------- Original Message --------
Subject: stream APIs
Date: Wed, 20 Jun 2012 13:58:19 +0300
From: Constantin Asofiei <ca@goldencode.com>
Reply-To: ca@goldencode.com
To: Greg Shah <ges@goldencode.com>
CC: Faulhaber, Eric <ecf@goldencode.com>

Greg,

I've sent an update for review some time ago (proposed update [ca_upd20120501a.zip]), which modifies the stream-related APIs emitted by the conversion code from assign... to:

UnnamedStreams.openTerminal[In|Out|Both](...)
UnnamedStreams.openProcess[In|Out|Both](...)
UnnamedStreams.openFIle[In|Out|Both](...)

and

Stream.openTerminal(...)
Stream.openProcess()
Stream.openFile(...)

Thanks,
Constantin

#5 - 10/23/2012 02:38 PM - Constantin Asofiei

Before moving on with these changes, we should test this update in 4GL.

#6 - 10/23/2012 02:39 PM - Greg Shah

Yes, exactly right. Some other feedback is below.

The code and approach looks fine with the following question:

1. Now that we can test the 4GL, please check to ensure that the issue is a P2J-specific problem.

2. I wonder if StreamWrapper.openFile() is correct. It closes the **existing** output stream (steam.closeOut()) ONLY IF the **new** stream is opened for writing. Likewise, it closes the **existing** input stream (steam.closeIn()) ONLY IF the **new** stream is opened for reading. But what about if the existing stream is open for reading and the new stream is opened for writing. I wonder if this is how the 4GL works, or if instead, whatever is previously opened is going to be closed (or flushed if it is the same file)?

3. I'm not sure if the same issue exists for UnnamedStreams.openFile{In,Out}. I suspect it is OK, but let's test the 4GL to make sure.

4. The UnnamedStreams copyright date needs to be set to 2012.

5. Did this ever pass regression testing in Majic?

#7 - 01/11/2013 10:53 AM - Constantin Asofiei

While Costin worked on #1626, he saw that the "attempt to write to closed stream" message does not appear properly when attempting to write to stream opened for input. The problem is with StreamWrapper.assign, which sets both the inOk and outOk flags to true - was this by intent ? IMO, for input streams outOk should be false (and viceversa for output streams). The change should be:

```
inOk = stream.isIn();
outOk = stream.isOut();
```

I ask this here because the StreamWrapper.assign was not changed by my work on streams, so if there is a problem and Costin fixes it in current StreamWrapper.assign, I don't want to lose it when I merge this into main trunk.

#8 - 01/11/2013 12:14 PM - Greg Shah

I don't know of any reason for the current implementation of StreamWrapper.assign(). The proposed approach seems to be a good improvement. I'm OK with it, either as part of this update or as part of Costin's update.

Files

ca_upd20120501a.zip	84.8 KB	10/23/2012	Greg Shah
ca_upd20120719a.zip	84.3 KB	10/23/2012	Greg Shah