Runtime Infrastructure - Bug #5701

logfile names for -O outputToFile use the spawner pid instead of the Java (FWD client process) pid

09/30/2021 12:22 PM - Greg Shah

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

Description

Related issues:

Related to Runtime Infrastructure - Bug #5699: broken logfile name on Windows...	**Closed**	
Related to Runtime Infrastructure - Bug #5703: rationalize, standardize and s...	**Closed**	
Related to Runtime Infrastructure - Feature #7147: Make FWD log outputs consi...	**New**	
Related to Runtime Infrastructure - Bug #7236: Inconsistencies in stderr and ...	**New**	**03/30/2023**

History

#1 - 09/30/2021 12:35 PM - Greg Shah

- Related to Bug #5699: broken logfile name on Windows appserver clients (and probably other cases on Windows) added

#2 - 09/30/2021 12:35 PM - Greg Shah

When clientConfig/outputToFile is specified in the directory for an appserver process, the value is treated as a specification which can have multiple placeholder values. One of the placeholds is %pid%.

The -O <filename> is passed on the spawner command line and is honored in the spawner. Search spawn.c for output_file_name or winspawn.c for szOutputFile to see the code.

This spawner code replaces the %pid% placeholder with the spawner's pid. What we really want is the Java (FWD client's) pid in that filename. Using the spawner's pid is confusing. It is the parent process for the Java client, but that means that the admin must find the OS parent/child process relationship to determine which FWD client is associated with the log. Not good.

I'm trying to understand why the Java client can't handle this redirection itself. We already do something similar with the clientLog 4GL command line feature. Why not standardize this and defer the redirection until after the Java client starts? That would allow us to put the proper pid in. It would also eliminate the #5699 issue because the Java code for replacing the %pid% placeholder will allow us to drop the broken Windows spawner code.

#3 - 10/04/2021 07:09 AM - Greg Shah

- Related to Bug #5703: rationalize, standardize and simplify the client-side log file name configuration added

#4 - 11/29/2022 08:36 AM - Greg Shah

- Assignee set to Galya B

#5 - 12/01/2022 06:51 AM - Galya B

This is how I read the logic for the spawner (I'll be going with the Unix example):

ClientBuilder#localStart() builds a new process to run the spawn command, i.e. the spawner process. The spawner process then creates a child process (1) and exits. The child process (1) sets some ids, opens a pseudo terminal master and slave, forks its own child process (2), closes the pty slave (converting itself to a pure pty master) and waits for the child status. Child process (2) closes pty master (converting itself to a pty slave), redirects stdin/stdout to pty slave and conditionally overwrites that statement with output redirect to a file if the arg is provided with the spawn cmd. At that stage the pid placeholder is replaced - in the pty slave process with its (child process 2) pid.

I still need to go through the code for the Windows spawner.

Now since this logic is related to batch processes, running in background where no client driver seems to be instantiated in the Java code, what do we refer to as Java client if not the pty slave process?

#6 - 12/01/2022 07:08 AM - Constantin Asofiei

Galya Bogdanova wrote:

> ... running in background where no client driver seems to be instantiated in the Java code, what do we refer to as Java client if not the pty slave process?

The spawner will use NativeSecureConnection.command to determine the launch command for the Java FWD client for this batch process, and will execute that command. So there is a FWD Java client for the batch process.

#7 - 12/01/2022 09:12 AM - Galya B

- Status changed from New to WIP

Constantin Asofiei wrote:

> The spawner will use NativeSecureConnection.command to determine the launch command for the Java FWD client for this batch process, and will execute that command. So there is a FWD Java client for the batch process.

Spawner launchP2JClient creates JVM in its own process and calls NativeSecureConnection#command() to get the actual command, but then it destroys the JVM before executing the logic I've mentioned in the previous comment. The retrieved command itself seems to start another JVM and that answers my question. Thanks for the pointer.

Now that new JVM should be actually created by executing the java command (from ClientBuilder) **in the slave pty process**:

```
410.          /* Execute command */
411.          exit(execvp(command, argv));
```

I found this description of the execvp command:

> This has the effect of running a new program with the process ID of the calling process. Note that a new process is not started; the new process image simply overlays the original process image.

I'm actually not completely sure if that means java exec will not be creating a new process for the JVM. I need to investigate it further.

If it does the Java client (started by the java command, built by ClientBuilder) will be running with the id of the spawner -> child -> child process in Unix. It's the same process as the one doing stdout_redirect.

#8 - 12/01/2022 10:49 AM - Galya B

- File PID-Test.zip added

I created a simple test to verify execvp behaviour and uploaded it as zip. The test function in C prints the pid and runs the Java app, which prints its pid. Running the included compiled ./test-app confirms that spawning a java app with execvp creates a JVM in the same process as the caller.

Please don't mind the source files Text Editor formatting.

To me it seems the proper client pid is set to the file. Am I missing something here?

#9 - 12/02/2022 11:17 AM - Greg Shah

The test code doesn't help me understand the process tree/pid list/redirected output log name for the FWD client. Can you please use ps and add logging to the C and Java code as needed to record an example of the various processes and how the logfile name is created and post it here?

#10 - 12/05/2022 04:41 AM - Galya B

- File PID-Test-1.zip added

- File spawner-processes-unix-mode0.svg added

Greg Shah wrote:

> The test code doesn't help me understand the process tree/pid list/redirected output log name for the FWD client. Can you please use ps and add logging to the C and Java code as needed to record an example of the various processes and how the logfile name is created and post it here?

The intention of the test was to verify the process id in both the spawner process and the spawned JVM by printing it from both places and thus confirming execvp behavior.

I updated slightly the example to made it closer to the original - the log file is renamed in the spawner and stdout is redirected. It's still just a small cut from the whole FWD spawner logic. I haven't tried to reproduce the whole process tree in this piece.

The uploaded image shows the whole spawner logic in Unix, while the test code simulates vaguely only the PTY slave <-> Java client (JVM) segment.

I'll update next with the spawning process in Windows.

#11 - 12/05/2022 06:18 AM - Galya B

- File spawner-processes-win.png added

The Windows spawner is a lot more simple than the UNIX one in the process creation part (more complicated in the security context part). I've attached an illustration.

As I've expected the actual bug can be reproduced in Windows only, because renaming is done in a different process. Currently we pass in the output file name with the STARTUPINFOW for CreateProcessWithTokenW/CreateProcessW and it uses the parent process id.

To fix it I would try the following approach - in the STARTUPINFOW **inherit** StdOutput handles from the parent process, create the child process, in the parent process format the log file name with the child pid and redirect the parent handlers to that file. Next I'll upload the code for the win test.

#12 - 12/05/2022 09:11 AM - Greg Shah

The approach seems reasonable.

Eugenie: Do you have any thoughts, questions or concerns?

#13 - 12/05/2022 09:29 AM - Eugenie Lyzenko

Greg Shah wrote:

> The approach seems reasonable.
>
> Eugenie: Do you have any thoughts, questions or concerns?

I need to see the proposed code changes for winspawn.c to make conclusion. Also we will have to try the change to ensure there are no regressions in Windows spawner.

#14 - 12/07/2022 07:07 AM - Greg Shah

From Galya, via email:

> I haven't looked at the Java classes instantiated by the spawner, so let me ask if there is any reason why we cannot redirect output in Java clients themselves and remove the C logic for output files? I can think of several issues, but there might be something else as well. It will require to identify each client class (or if there is a base one) that can be passed to the spawner in the command built by ClientBuilder and then redirect System.out and System.err in its main method. First of all that would mean changes to more places in the code. Also I don't know if the Java process will have the same permissions to access the workingDir as the spawner. The code for getting pid cross-platform in Java 8 relies on reflection, not ideal and will probably have to be changed with future update of the JDK.

It is a reasonable question. For the 4GL compatibility, when we launch child processes (for example, to implement the 4GL OS-COMMAND statement) we know that the Java process builder support does not give us suitable tools to setup the process state that will work. The redirections are a big part of this (the parent process often must have write access to the STDIN and read access to the combined STDOUT/STDERR). Just as important is the pseudo-terminal state, signal processing and other OS level configuration. Although Java has improved process launching in recent versions (after Java 8), I believe it still doesn't provide the capabilities that we require.

On the other hand, the scenario you are talking about does not have these same requirements. The parent process is the spawner and I don't think

we need continued access to the redirected streams. On the other hand, we very much do need control over the working dir, environment, permissions and other process launching state. More importantly, the spawner process is needed for the core of the processing due to the requirement of using the OS-level APIs to start the child process with the correct OS account. I don't think we can do that directly in Java. If you find otherwise, we can consider it.

#15 - 12/15/2022 07:26 AM - Galya B

My proposal cannot actually work for Windows where CreateProcess creates an independent new process without any references. Explained here:

> The CreateProcess function creates a new process, which runs **independently** of the creating process. However, **for simplicity, the relationship is referred to as a parent-child** relationship.

There is no way to modify the handlers of the child from the parent process.
Basically we set some startup details and a startup command and let it go.

Some compile C with Cygwin where you can simulate fork() and this way get access to the new process (how it's done for the Linux spawner where things work). But I saw here that we don't use Cygwin runtime and I'm not sure if it works if only used compile time.

I'm thinking of only two possible solutions (still not explored):
1. Do it in Java - only for that specific case of a spawned background process, where outputToFile is defined. Then redirect System.out in the ClientDriver like we do with System.err in debug.
2. Run the cmd in the spawner process itself. This looks like a more dangerous exercise, since I don't fully understand the win spawn process. But I can work on it.

Any thoughts on these?

#16 - 12/15/2022 09:00 AM - Greg Shah

> Run the cmd in the spawner process itself. This looks like a more dangerous exercise, since I don't fully understand the win spawn process. But I can work on it.

I don't think we can do this. The spawner process is privileged and the executed code must not have elevated privileges.

#17 - 12/15/2022 01:17 PM - Galya B

- File outputToFile-in-java.diff added

I'm attaching a diff for a demo implementation of how %pid% replace and stdout redirect can work in Java for background / batch processes with defined outputToFile config in directory.xml, so that it will be consistent between OSes. This is not the final code of such implementation, but just a snippet for testing purposes. I need an ABL procedure that outputs to outputToFile to test it out. If you know what I should be looking for, please give me a hint. This diff can be tested by recompiling and installing the spawner with function call stdout_redirect() commented out.

The conclusion of my analysis is that we should either implement that as a temporal solution to the issue, or close the task and solve the problem on the bigger scale of #3853 and #5703, where that log file may become redundant.

#18 - 12/15/2022 02:56 PM - Greg Shah

> I need an ABL procedure that outputs to outputToFile to test it out.

I think you can try any 4GL code which uses DISPLAY, PUT or similar output statement without otherwise specifying the target stream.

Something as simple as DISPLAY "hello". will probably do it. You should also check a child process that generates output to stdout (e.g. OS-COMMAND echo "child-process hello".).

#19 - 12/16/2022 02:30 AM - Galya B

Greg Shah wrote:

> I think you can try any 4GL code which uses DISPLAY, PUT or similar output statement without otherwise specifying the target stream.
>
> Something as simple as DISPLAY "hello". will probably do it. You should also check a child process that generates output to stdout (e.g. OS-COMMAND echo "child-process hello".).

I have exactly that - a simple procedure with DISPLAY, but I've never seen anything in that log file. That is with the new testcases scheduled appserver_process and p2j-entry properly configured. There is no issue in any server / client logs. The file is created. It's just empty. Not sure what I'm missing.
Also the server log shows (AppServerLauncher:INFO) Appserver 'app_server' was started successfully (4945ms)!
P.S. I also just tested OS-COMMAND echo "child-process hello". and still got nothing in the log file.

#20 - 12/16/2022 02:52 AM - Galya B

I have another idea of how to solve the pid issue on Windows.

The Windows spawner creates the new process with all the needed attributes as it is now, but the cmd is executing another C compiled file. That's where we set a stdout handler to the properly renamed filename and execute the cmd to spawn JVM received as an argument in the same process.

It will not add much complexity and will be a change only for Windows.

Do you think it's acceptable?

#21 - 12/16/2022 03:01 AM - Galya B

Galya Bogdanova wrote:

> That's where we set a stdout handler to the properly renamed filename and execute the cmd to spawn JVM received as an argument in the same process.

Actually it doesn't even have to set any handlers. Just rename the file and execute the java cmd with > outputToFile

#22 - 12/16/2022 05:48 AM - Galya B

Galya Bogdanova wrote:

> The Windows spawner creates the new process with all the needed attributes as it is now, but the cmd is executing another C compiled file. That's where we set a stdout handler to the properly renamed filename and execute the cmd to spawn JVM received as an argument in the same process.

This won't work either. I thought system() in Windows can replicate the behavior of execvp() in Unix, but nothing similar is possible, so the java cmd will always be launched in a separate process and its pid will be available only to the Java app.

#23 - 12/16/2022 07:24 AM - Greg Shah

Please focus on #5703 first and pause work on this task.

#24 - 12/21/2022 03:55 AM - Galya B

I'm not working on this task at the moment, but I randomly got an idea and just want to document it here, since we might need to keep this output file (in Progress it's a thing for batch). It might be easier to find the right log by process create datetime instead of pid of the spawner, so changing the placeholder (or adding a new) might help for Windows.

#25 - 01/26/2023 08:01 AM - Galya B

- File pid-in-outputToFile-fix.diff added

Find attached a good solution to the wrong pid in outputToFile name. It simplifies the spawner logic by removing a big chunk of it related to renaming a file. The win version needs to be modified as well, if the solution is approved. Having the redirect in Java also solves the issue of printing symbols in the beginning and at the end of the file (ref #6921#note-71).

#26 - 01/26/2023 08:05 AM - Galya B

- Status changed from WIP to Review

- % Done changed from 0 to 80

#27 - 01/26/2023 08:15 AM - Greg Shah

Eugenie: Please review.

#28 - 01/26/2023 10:35 AM - Eugenie Lyzenko

Greg Shah wrote:

> Eugenie: Please review.

The change looks good for me.

#29 - 01/27/2023 05:47 AM - Galya B

Starting appserver as scheduled batch initializes ThinClient, BatchDriver, OutputManager, ErrorWriterBatch, calls StandardServer#standardEntry() where AgentPool and Agent are started to listen to commands and then the client receives false as response from the server and quits. The client never uses BatchPrimitives to update the screen and write to stdout, that's why handles as DISPLAY or MESSAGE VIEW AS ALERT-BOX have no output in the file in appserver mode.

Is this the expected behavior? I've asked about it before. The file outputToFile in appserver mode is empty. I think I've seen only messages generated in the client (not from converted procedures) to be written to it. Is there someone who actually uses it successfully?

#30 - 01/27/2023 09:09 AM - Galya B

- Status changed from Review to Test

- File patch-pid-in-outputToFile-plus-win-fix.diff added

- % Done changed from 80 to 100

I'm uploading the latest diff that includes the win changes. They are committed to 5701a as well. I'll be testing thoroughly next.

#31 - 01/27/2023 09:53 AM - Constantin Asofiei

Galya, I've tested in OpenEdge and only MESSAGE statement (with or without VIEW-AS ALERT-BOX) end up in the agent log. I can't make the DISPLAY or PUT SCREEN emit to the log.

In the FWD Client, the MESSAGE statement ends up in Window.message - what is FWD doing in this code, if is reached, in the Agent case?

#32 - 01/27/2023 09:59 AM - Greg Shah

> Galya, I've tested in OpenEdge and only MESSAGE statement (with or without VIEW-AS ALERT-BOX) end up in the agent log. I can't make the DISPLAY or PUT SCREEN emit to the log.

The agent log still gets all the "internal" (4GL runtime generated logging events) log-manager stuff right? How do we handle that today? I recall we didn't handle this yet.

#33 - 01/27/2023 10:38 AM - Constantin Asofiei

Greg Shah wrote:

> > Galya, I've tested in OpenEdge and only MESSAGE statement (with or without VIEW-AS ALERT-BOX) end up in the agent log. I can't make the DISPLAY or PUT SCREEN emit to the log.
>
> The agent log still gets all the "internal" (4GL runtime generated logging events) log-manager stuff right?

You mean error messages or such? Yes, these end up in the agent log in OpenEdge and in FWD, too, if I'm recalling right.

If you mean LOG-MANAGER:WRITE-MESSAGE, in OpenEdge this ends up in the default agent log, if LOGFILE-NAME is not set.

#34 - 01/27/2023 10:44 AM - Greg Shah

I meant both cases. It sounds like we are OK there, if I understand your points.

Do we have the same control over the log filename for these sessions as is possible in OE? I recall we had some limitations on our agent logging, but I don't recall what those limitations are (or were).

#35 - 01/30/2023 03:25 AM - Galya B

Greg, Constantin, I'll need some clarifications here:
1. What do you refer to as agent logs in FWD? outputToFile is the only one I know of.
2. How do you test (run) appserver agents in the OpenEdge env? Can you give me some cmds to run.
3. Can you confirm outputToFile logs output as expected?

It's fine if you don't have the answer to 3. since that is what we're discussing here.

Also let's get LOG-MANAGER out of the picture for now. In the discussed cases let's have it disabled and let's try to reproduce 4GL behavior.

I haven't tested appserver runs in OpenEdge, so there might be a difference in the agent output compared to batch output, but nevertheless something needs to go out to this outputToFile file in our implementation and it's not. I can't see it used in the FWD code, nor in my test runs.

I'm explain myself better - there is a difference in FWD in the output between batch executed as a standalone client from command line and as scheduled service (aka appserver_process). outputToFile is a directory.xml configuration that applies **only** to batch running as a scheduled service. As far as I understand it outputToFile is expected to replace the stdout redirect from running batch cmd line like that > output.txt. If not can you point me to some documentation to understand better what we're reproducing with outputToFile.

Standalone running batch works as expected, similar to what we have in 4GL - any MESSAGE or DISPLAY statement is saved to the file.

As I've said already I'm not seeing this behavior with appserver service (scheduled batch) and it's clear from the code as well when debugging that nothing is going back from the agent to the client process to be output.

Please give me more details.

#36 - 01/30/2023 05:05 AM - Constantin Asofiei

Galya Bogdanova wrote:

> Greg, Constantin, I'll need some clarifications here:
> 1. What do you refer to as agent logs in FWD? outputToFile is the only one I know of.

outputToFile is for STDOUT, which is the equivalent for OpenEdge's Agent default log or batch program's STDOUT. FWD also has a STDERR log.

> 2. How do you test (run) appserver agents in the OpenEdge env? Can you give me some cmds to run.

What machine are you using? If is a customer one, please post to #6921.

> 3. Can you confirm outputToFile logs output as expected?

Yes, in FWD it works as expected for appserver agents, I've tested with 6129b.

How are you testing the appserver in FWD? Are you just starting the appservers? Keep in mind that outputToFile can be configured for either batch or agent FWD clients, so if you can test batch programs and it works there, then is OK.

Please test this in a batch program, started both via command line and ./server.sh -b or directory.xml scheduler:

```
message "this is error1".  // logged in both
message "this is error2" view-as alert-box. // logged in both

display "display is emitted only in batch". // this is logged in batch, but not in appserver agent
put screen row 1 col 1 "put is not emitted". // this is not logged in either batch or appserver agent

// in appserver agent, there is always a default log, and this is written to that log
// for batch, if no log is opened, 'Cannot write message to log, as there is no log open (14332)' is shown on screen
log-manager:write-message("this is logged").
```

```
def var v as handle.
do on error undo, leave:
   delete object v. // logged in both
end.
```

#37 - 01/30/2023 06:40 AM - Galya B

So configuration-wise appserver turns into batch if I remove <node-attribute name="appserver" value="app_server"/> from directory.xml:

```
<node class="container" name="">
  <node class="container" name="security">
    <node class="container" name="accounts">
      <node class="container" name="processes">
        <node class="process" name="appserver_process">
          <node-attribute name="appserver" value="app_server"/>
```

Then there is output indeed.

If the same procedure is run as appserver, it doesn't output anything. So is this the expected behavior? In that case is it expected that only rest / soap agents write to outputToFile, when endpoints are hit? How is outputToFile supposed to be used with appservers?

#38 - 01/30/2023 07:24 AM - Constantin Asofiei

Galya, please describe how you are running a program on the appserver. A batch program executes its 4GL program immediately. In the appserver case, you need another 4GL program to connect to the appserver and run the 4GL program.

#39 - 01/31/2023 03:56 AM - Galya B

Constantin Asofiei wrote:

> Galya, please describe how you are running a program on the appserver. A batch program executes its 4GL program immediately. In the appserver case, you need another 4GL program to connect to the appserver and run the 4GL program.

I've used the new testcases template as it is, which supports appserver scheduled for immediate run. The procedure though didn't set an appserver and no agents were connected. My question is if we expect the appserver procedure (the server-side) to produce output to outputToFile the same way as a scheduled batch? Since StandardServer always returns false (for running) to the client JVM right away, it will never do.

P.S. I'll try to test the original behavior in OpenEdge soon, but if we already have it described somewhere it would be helpful to know.

#40 - 01/31/2023 04:47 AM - Galya B

- File 5701a-r14485.diff added

I've tested both Linux and Windows spawner. Here are the final changes in the .diff file, also committed to 5701a.

Greg, please comment if a final review is needed and when I can merge to trunk.

#41 - 01/31/2023 03:53 PM - Greg Shah

Code Review Task Branch 5701a Revisions 14484 and 14485

1. In spawn.c:launchP2JClient, the memset(arguments[length + 1], '\0', len); should use len * sizeof(char) instead of len. The winspawn.c has the same issue.

2. Please add the history number to your history entries. For example, use 031 for ClientCore, 005 for ProcessClientBuilder and so forth.

3. Please add javadoc for ClientCore.setOutputToFile().

#42 - 01/31/2023 03:54 PM - Greg Shah

Eugenie: Please review branch 5701a.

#43 - 01/31/2023 04:46 PM - Eugenie Lyzenko

Greg Shah wrote:

> Eugenie: Please review branch 5701a.

I have no objections, in addition to history entries it is required to change Year to 2023.

Also in winspawn.c I would add the initial setup for module static var szOutputFile[MAX_SIZE] = "". But it is just a style preferences I guess, the current version should also work.

#44 - 02/01/2023 02:43 AM - Galya B

Greg Shah wrote:

> Code Review Task Branch 5701a Revisions 14484 and 14485
>
> 1. In spawn.c:launchP2JClient, the memset(arguments[length + 1], '\0', len); should use len * sizeof(char) instead of len. The winspawn.c has the same issue.

https://en.cppreference.com/w/cpp/language/sizeof says:

> The following sizeof expressions always evaluate to 1:

sizeof(char)
sizeof(signed char)
sizeof(unsigned char)

I found the same statement in other sources as well.

Maybe what we're looking for according to Stackoverflow is sizeof(char*)?

> char is a character and sizeof(char) is defined to be 1. (N1570 6.5.3.4 The sizeof and _Alignof operators, paragraph 4)
>
> char* is a pointer to a character and sizeof(char*) depends on the environment. It is typically 4 in 32-bit environment and 8 in 64-bit environment.

Also my tests show that the string is not cut, but on other env it might not work.

What about changing it the linux version from:

```
1021      int len = strlen(output_file_name) + 1;
1022      arguments[length + 1] = (char*) malloc(len * sizeof(char));
1023      memset(arguments[length + 1], '\0', len);
1024      strcpy(arguments[length + 1], output_file_name);
```

to:

```
1021      int size = (strlen(output_file_name) + 1) * sizeof(char*);
1022      arguments[length + 1] = (char*) malloc(len * sizeof(char*));
1023      memset(arguments[length + 1], '\0', size);
1024      strcpy(arguments[length + 1], output_file_name);
```

Please note the change on line 1022. It should apply to line 1059 as well. The changes will be applied to win as well.

About the other points, I'm on it.

#45 - 02/01/2023 03:02 AM - Galya B

Eugenie Lyzenko wrote:

> I have no objections, in addition to history entries it is required to change Year to 2023.

It's 2023 for the entries related to 5701, but there were no history entry numbers and maybe you saw the ones for the previous task. When I add the numbers it will look fine.

> Also in winspawn.c I would add the initial setup for module static var szOutputFile[MAX_SIZE] = "". But it is just a style preferences I guess, the current version should also work.

It feels weird to a Java developer to initialize an array that way and there is a mixture of both styles in the file, but I'll make the change for this one.

#46 - 02/01/2023 06:01 AM - Eugenie Lyzenko

Galya Bogdanova wrote:

> Eugenie Lyzenko wrote:
>
> > I have no objections, in addition to history entries it is required to change Year to 2023.
>
> It's 2023 for the entries related to 5701, but there were no history entry numbers and maybe you saw the ones for the previous task. When I add the numbers it will look fine.

I meant the following:

```
** Copyright (c) 2013-2021, Golden Code Development Corporation.
...
**     EVL 20210519 Fix for segmentation fault in strcmp() library call.  Added new error codes to inform
**                  the caller about system errors.
```

When you add new history entry with date (2023MMDD), and you should mark your change the year on the top of the header should match new entry:

```
** Copyright (c) 2013-2023, Golden Code Development Corporation.
...
** 014 GBB 20230201 Some text.
```

We need to avoid any modifications without history log. It will be very difficult to understand what and when was changed otherwise.

#47 - 02/01/2023 06:04 AM - Galya B

Eugenie Lyzenko wrote:

> When you add new history entry with date (2023MMDD), and you should mark your change the year on the top of the header should match new entry:

Did you review file 5701a-r14485.diff ? There are 4 source files modified and all of them have updated year.

#48 - 02/01/2023 09:25 AM - Eugenie Lyzenko

Galya Bogdanova wrote:

> Eugenie Lyzenko wrote:
>
> > When you add new history entry with date (2023MMDD), and you should mark your change the year on the top of the header should match new entry:
>
> Did you review file 5701a-r14485.diff ? There are 4 source files modified and all of them have updated year.

Yes, I did. src/native/spawn.c has neither history nor year update. src/native/winspawn.c has last update entry as of GBB 20221209.

#49 - 02/01/2023 09:43 AM - Galya B

- File 5701a-r14485.diff-6.png added

- File 5701a-r14485.diff-7.png added

- File 5701a-r14485.diff-5.png added

- File 5701a-r14485.diff-4.png added

- File 5701a-r14485.diff-3.png added

- File 5701a-r14485.diff-2.png added

- File 5701a-r14485.diff-1.png added

Eugenie Lyzenko wrote:

> Yes, I did. src/native/spawn.c has neither history nor year update. src/native/winspawn.c has last update entry as of GBB 20221209.

There are 4 .diff files attached to the task, the latest is called 5701a-r14485.diff and below is its content.

I downloaded it from the task 5 times or more and it's the same every time. Is it what you're seeing?

Open 5701a-r14485.diff ~/Music Save

```
=== modified file 'src/com/goldencode/p2j/main/ClientCore.java'
--- src/com/goldencode/p2j/main/ClientCore.java 2022-12-14 16:21:35 +0000
+++ src/com/goldencode/p2j/main/ClientCore.java 2023-01-27 13:53:55 +0000
@@ -2,7 +2,7 @@
 ** Module   : ClientCore.java
 ** Abstract : core client loop which handles init and connects to server
 **
-** Copyright (c) 2005-2022, Golden Code Development Corporation.
+** Copyright (c) 2005-2023, Golden Code Development Corporation.
 **
 ** -#- -I- --Date-- ----------------------------------Description---------------------------------
 ** 001 GES 20110928 Moved core logic here from ClientDriver so that the same
@@ -62,6 +62,7 @@
 **     CA  20220918 Added socket trust store filename and password to the ClientParameters.
 **     GBB 20221209 Path to debug log file to be read from client bootstrap configs.
 **                  Passing arg to spawner for path to stderr log file for background processes.
+**     GBB 20230127 Fixing pid in outputToFile name by replacing it in Java code.
 */
 /*
 ** This program is free software: you can redistribute it and/or modify
@@ -118,6 +119,7 @@

 package com.goldencode.p2j.main;

+import java.io.*;
 import java.util.*;
 import java.util.function.*;
 import java.util.logging.*;
@@ -186,6 +188,11 @@
    public static void start(BootstrapConfig config, boolean file, boolean single, Supplier<String> diag)
    throws Exception
    {
+      // Load native library
+      loadNativeLibrary();
+
+      setOutputToFile(config);
+
       String  uuid     = config.getString("server", "spawner", "uuid", null);
       boolean dbg      = config.getBoolean("logging", "debug", "enable", false);
       String  referrer = config.getString("web", "referrer", "url", null);
@@ -196,9 +203,6 @@
       // iterations of the core loop
       ScreenDriver<?> driver = (uuid == null) ? null : processTemporaryClient(uuid, config);

-      // Load native library
-      loadNativeLibrary();
-
       while (running)
       {
          Object          context = null;
@@ -434,6 +438,25 @@
       return isStop;
    }

+   private static void setOutputToFile(BootstrapConfig config)
+   {
+      String outputToFile = config.getString("server", "clientConfig", "outputToFile", null);
+      if (outputToFile == null)
+      {
+         return;
+      }
```

```
+      }
+
+      try
+      {
+         outputToFile = outputToFile.replace("%pid%", String.valueOf(getPid()));
+         System.setOut(new PrintStream(new FileOutputStream(outputToFile), true));
+      }
+      catch (Throwable t)
+      {
+         System.err.println("ClientCore: Unable to redirect stdout to outputToFile " + outputToFile);
+      }
+   }
+
    /**
     * Loads native library and performs required initial steps for P2J library to working.
     */

=== modified file 'src/com/goldencode/p2j/main/ProcessClientBuilder.java'
--- src/com/goldencode/p2j/main/ProcessClientBuilder.java       2017-04-01 23:33:34 +0000
+++ src/com/goldencode/p2j/main/ProcessClientBuilder.java       2023-01-27 13:56:11 +0000
@@ -2,7 +2,7 @@
 ** Module   : ProcessClientBuilder.java
 ** Abstract : build and start a platform process for P2J process.
 **
-** Copyright (c) 2014-2017, Golden Code Development Corporation.
+** Copyright (c) 2014-2023, Golden Code Development Corporation.
 **
 ** -#- -I- --Date-- --------------------------------Description----------------------------------
 ** 001 CA  20140206 First version.
@@ -10,6 +10,7 @@
 ** 003 MAG 20140722 Implemented the remote launcher.
 ** 004 GES 20160219 Moved code to the parent class allowing reuse of the native secure connection
 **                  setup processing.
+**     GBB 20230127 Fixing pid in outputToFile name by passing it as config to spawned client.
 */
 /*
 ** This program is free software: you can redistribute it and/or modify
@@ -132,9 +133,9 @@
       if (outputToFile != null)
       {
          cmd.add("-O");
-         cmd.add(outputToFile);
+         cmd.add("server:clientConfig:outputToFile=" + outputToFile);
       }

       return cmd;
    }
-}
\ No newline at end of file
+}

=== modified file 'src/native/spawn.c'
--- src/native/spawn.c  2021-05-19 21:40:23 +0000
+++ src/native/spawn.c  2023-01-31 09:34:16 +0000
@@ -2,7 +2,7 @@
 ** Module   : spawn.c
 ** Abstract : a simple tool able to spawn a process on an existing user account.
 **
-** Copyright (c) 2013-2021, Golden Code Development Corporation.
+** Copyright (c) 2013-2023, Golden Code Development Corporation.
 **
```

Open

5701a-r14485.diff
~/Music

Save

```
 ** -#- -I- --Date-- -------------------------------Description---------------------------------
 ** 001 MAG 20131127 First version.
@@ -27,6 +27,8 @@
 **     IAS 20210325 Added addional parameters for command
 **     EVL 20210519 Fix for segmentation fault in strcmp() library call.  Added new error codes to inform
 **                  the caller about system errors.
+**     GBB 20230131 outputToFile passed in as arg to create client process command.
+**                  stdout redirect in Java.
 */
 /*
 ** This program is free software: you can redistribute it and/or modify
@@ -162,9 +164,6 @@
    #define ALLPERMS (S_ISUID|S_ISGID|S_ISVTX|S_IRWXU|S_IRWXG|S_IRWXO)/* 07777 */
 #endif

-/* prototypes */
-void stdout_redirect();
-
 /* Global data */
 struct passwd *pw;
 int extra_logging = 0;
@@ -175,6 +174,7 @@

 /* Output file name */
 char output_file_name[MAX_FILENAME];
+int output_to_file_opt_found = 0;

 /**
  * Use syslog to audit the execution of child processes.
@@ -400,12 +400,6 @@
             logMessage(message_buffer);
             exit(audit(ERR_SPAWN_CHDIR_WD, "chdir(workdir)"));
          }
-
-         /* Redirect STDOUT if required */
-         if (output_file_name[0] != 0)
-         {
-            stdout_redirect();
-         }

          /* Execute command */
          exit(execvp(command, argv));
@@ -1018,9 +1012,17 @@
    /* make it 1-based, so it follows the structure of the command line arguments
       (0 is program name); also, one more position is needed for execvp, which requires for the
       arguments to be null-terminated. */
-   char **arguments = (char**) malloc((length + 2) * sizeof(char*));
+   char **arguments = (char**) malloc((length + 2 + output_to_file_opt_found) * sizeof(char*));
    arguments[0] = NULL;
-   arguments[length + 1] = NULL;
+   arguments[length + 1 + output_to_file_opt_found] = NULL;
+
+   if (output_to_file_opt_found == 1)
+   {
+      int len = strlen(output_file_name) + 1;
+      arguments[length + 1] = (char*) malloc(len * sizeof(char));
+      memset(arguments[length + 1], '\0', len);
+      strcpy(arguments[length + 1], output_file_name);
+   }

    int i;
```

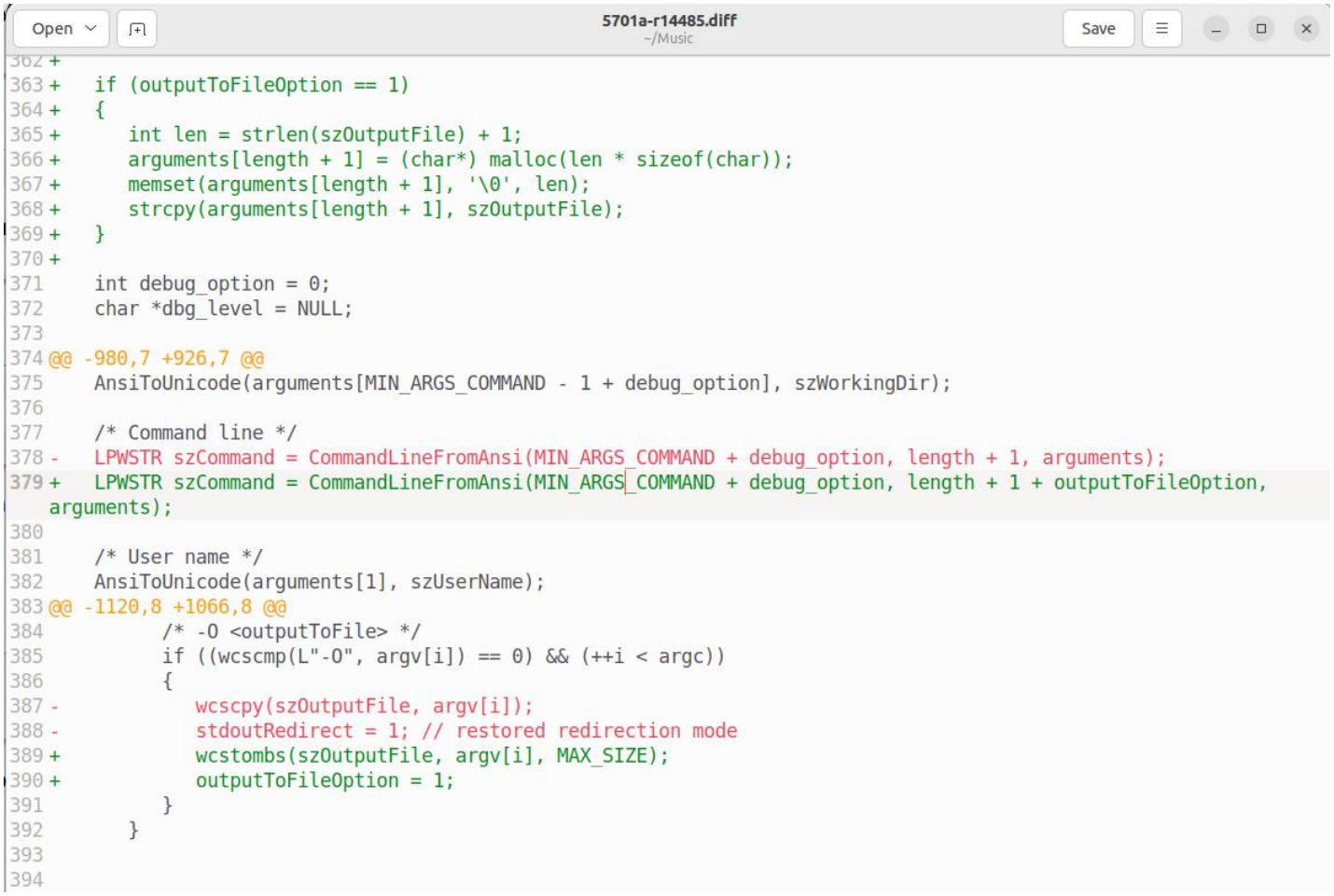
Open | 5701a-r14485.diff ~/Music | Save

```
    int i;
    for (i = 0; i < (int) length; i++)
@@ -1151,38 +1153,6 @@
 }

 /**
- * Redirect STDOUT to an output file.
- */
-void stdout_redirect()
-{
-   int output_fd;
-   char pid[32];
-
-   /* convert pid to string */
-   sprintf(pid, "%u", getpid());
-
-   /* replace %pid% placeholder */
-   char* filename = replace_str(output_file_name, "%pid%", pid);
-
-   // Open file
-   output_fd = open(filename, O_WRONLY | O_CREAT | O_APPEND, 0644);
-
-   if (output_fd == -1)
-   {
-      fprintf(stderr, "Cannot open file %s\n", filename);
-   }
-   else
-   {
-      /* Redirect STDOUT to file */
-      if (dup2(output_fd, STDOUT_FILENO) == -1)
-      {
-         close(output_fd);
-         fprintf(stderr, "Cannot redirect STDOUT to file %s\n", filename);
-      }
-   }
-}
-
-/**
  * Main function which allows both password and P2J server-style authentication.
  * <p>
  * For P2J server-style authentiation, the following files need to exist in current directory:
@@ -1335,6 +1305,7 @@
          if ((strcmp("-O", argv[i]) == 0) && (++i < argc))
          {
            strcpy(output_file_name, argv[i]);
+           output_to_file_opt_found = 1;
          }
       }


=== modified file 'src/native/winspawn.c'
--- src/native/winspawn.c       2022-12-09 10:28:29 +0000
+++ src/native/winspawn.c       2023-01-31 09:33:05 +0000
@@ -2,7 +2,7 @@
 ** Module   : winspawn.c
 ** Abstract : a simple tool able to spawn a process on an existing user account.
 **
-** Copyright (c) 2013-2022, Golden Code Development Corporation.
+** Copyright (c) 2013-2023, Golden Code Development Corporation.
 **
 ** #   I    Date                                   Description
```

Open | 5701a-r14485.diff ~/Music | Save

```
** -#- -I- --Date-- -------------------------------Description---------------------------------
** 001 MAG 20131130 First version.
@@ -27,8 +27,10 @@
**                  to CreateProcessWithToken.
**     EVL 20220413 Fixed the issues with processes starting on behalf of different users.  Some additional
**                  improvements for common desktop name.
-** 010 GBB 20221209 Fix for stdout file name, incorrect calculation of the prefix length
+**     GBB 20221209 Fix for stdout file name, incorrect calculation of the prefix length
**                  when %pid% placeholder replaced.
+**     GBB 20230131 outputToFile passed in as arg to create client process command.
+**                  stdout redirect in Java.
*/
/*
** This program is free software: you can redistribute it and/or modify
@@ -132,9 +134,8 @@

unsigned char key1[KEY_SIZE + 1], key2[KEY_SIZE + 1], key3[KEY_SIZE + 1];

-/* STDOUT redirection flag. Default NO redirection */
-int stdoutRedirect = 0;
-WCHAR szOutputFile[MAX_SIZE] = L"";
+int outputToFileOption = 0;
+char szOutputFile[MAX_SIZE];

WCHAR szUserProfile[MAX_SIZE] = L"";
WCHAR szCommandLine[MAX_SIZE] = L"";
@@ -510,58 +511,6 @@
}

/**
-* Redirect STDOUT to an output file.
-*
-* @param   szWorkingDir
-*          Working directory.
-* @param   szOutputFile
-*          The name of the file where the STDOUT is redirected.
-*/
-void stdout_redirect(LPWSTR szWorkingDir, LPWSTR szOutputFile)
-{
-   HANDLE hFile;
-   WCHAR pid[32];
-
-   /* Goto working directory */
-   if (!SetCurrentDirectoryW(szWorkingDir))
-   {
-      DisplayError(L"SetCurrentDirectoryW");
-   }
-
-   /* convert pid to string */
-   swprintf(pid, sizeof(pid), L"%lu", GetCurrentProcessId());
-
-   /* replace %pid% placeholder */
-   LPWSTR szFileName = replace_str(szOutputFile, L"%pid%", pid);
-
-   SECURITY_ATTRIBUTES sa;
-   sa.nLength = sizeof(sa);
-   sa.lpSecurityDescriptor = NULL;
-   sa.bInheritHandle = TRUE;
-
-   /* create file */
-   hFile = CreateFileW(szFileName,                         // file name
```

```
-   hFile = CreateFileW(szFileName,                     // file name
-                       FILE_APPEND_DATA,               // open for writing
-                       FILE_SHARE_WRITE | FILE_SHARE_READ, // allow multiple readers
-                       &sa,                            // security
-                       OPEN_ALWAYS,                    // open or create
-                       FILE_ATTRIBUTE_NORMAL,          // normal file
-                       NULL);                          // no template
-
-   /* check for errors */
-   if (hFile == INVALID_HANDLE_VALUE)
-   {
-      DisplayError(L"CreateFileW");
-   }
-   else
-   {
-      /* redirect */
-      si.hStdOutput = hFile;
-      si.dwFlags |= STARTF_USESTDHANDLES;
-   }
-}
-
-/**
 * Spawn a child process on the current logon account.
 *
 * @param   szWorkingDir
@@ -579,12 +528,6 @@
    si.wShowWindow = SW_HIDE;
    // since Windows Vista we have to specify desktop session explicitly here
    si.lpDesktop = desktopDefault;
-
-   /* Redirect STDOUT if required */
-   if (stdoutRedirect)
-   {
-      stdout_redirect(szWorkingDir, szOutputFile);
-   }

    if (!CreateProcessW(NULL,             // application name
                        szCommand,        // command line
@@ -681,12 +624,6 @@
       }
       szWorkingDir = szUserProfile;
    }
-
-   /* Redirect STDOUT if required */
-   if (stdoutRedirect)
-   {
-      stdout_redirect(szWorkingDir, szOutputFile);
-   }

    if (extra_logging > 0)
    {
@@ -914,9 +851,18 @@
    /* make it 1-based, so it follows the structure of the command line arguments
       (0 is program name); also, one more position is needed for execvp, which requires for the
       arguments to be null-terminated. */
-   char **arguments = (char**) malloc((length + 2) * sizeof(char*));
+   char **arguments = (char**) malloc((length + 2 + outputToFileOption) * sizeof(char*));
    arguments[0] = NULL;
-   arguments[length + 1] = NULL;
+   arguments[length + 1 + outputToFileOption] = NULL;
+
```




```
362 +
363 +   if (outputToFileOption == 1)
364 +   {
365 +      int len = strlen(szOutputFile) + 1;
366 +      arguments[length + 1] = (char*) malloc(len * sizeof(char));
367 +      memset(arguments[length + 1], '\0', len);
368 +      strcpy(arguments[length + 1], szOutputFile);
369 +   }
370 +
371     int debug_option = 0;
372     char *dbg_level = NULL;
373
374 @@ -980,7 +926,7 @@
375     AnsiToUnicode(arguments[MIN_ARGS_COMMAND - 1 + debug_option], szWorkingDir);
376
377     /* Command line */
378 -   LPWSTR szCommand = CommandLineFromAnsi(MIN_ARGS_COMMAND + debug_option, length + 1, arguments);
379 +   LPWSTR szCommand = CommandLineFromAnsi(MIN_ARGS_COMMAND + debug_option, length + 1 + outputToFileOption,
    arguments);
380
381     /* User name */
382     AnsiToUnicode(arguments[1], szUserName);
383 @@ -1120,8 +1066,8 @@
384           /* -O <outputToFile> */
385           if ((wcscmp(L"-O", argv[i]) == 0) && (++i < argc))
386           {
387 -            wcscpy(szOutputFile, argv[i]);
388 -            stdoutRedirect = 1; // restored redirection mode
389 +            wcstombs(szOutputFile, argv[i], MAX_SIZE);
390 +            outputToFileOption = 1;
391           }
392        }
393
394
```

#50 - 02/01/2023 10:17 AM - Eugenie Lyzenko

Galya Bogdanova wrote:

> Eugenie Lyzenko wrote:
>
> > Yes, I did. src/native/spawn.c has neither history nor year update. src/native/winspawn.c has last update entry as of GBB 20221209.
>
> There are 4 .diff files attached to the task, the latest is called 5701a-r14485.diff and below is its content.
>
> I downloaded it from the task 5 times or more and it's the same every time. Is it what you're seeing?

OK. Now I've got the versions you mentioned.

It is 14486, not 14485, right? So far I have no notes for 14486. I think this is versions misunderstanding...

#51 - 02/01/2023 11:40 AM - Greg Shah

Lets consider this code from spawn.c:

```
   char **arguments = (char**) malloc((length + 2 + output_to_file_opt_found) * sizeof(char*));
   arguments[0] = NULL;
   arguments[length + 1 + output_to_file_opt_found] = NULL;

   if (output_to_file_opt_found == 1)
   {
      int len = strlen(output_file_name) + 1;
      arguments[length + 1] = (char*) malloc(len * sizeof(char));
      memset(arguments[length + 1], '\0', len);
      strcpy(arguments[length + 1], output_file_name);
   }
```

arguments is a char** which means "pointer to a char pointer". This is a technique in C (and C++) in which one allocates a region of memory for an array of pointers. The pointers can be accessed via array syntax (arguments[0] or arguments[length - 1]) or via direct pointer arithmetic.

Each pointer in this array, is a pointer to char data. That pointer references a separate block of memory that needs to be separately allocated (via malloc() or other means).

The reference to arguments[length + 1] in memset(arguments[length + 1], '\0', len); is passing the array element at length + 1 which is a char* (a pointer to char data). memset() is going to blast a bit pattern into the memory referenced by that block. That referenced memory was just allocated using arguments[length + 1] = (char*) malloc(len * sizeof(char));. The size of that block is len * sizeof(char).

When you use memset() to then overwrite that block with the '\0' char, you must do that with the proper size which is sizeof(char), the same thing used in the malloc().

I highly recommend the bible of C programming by Kernighan and Ritchie: The C Programming Language (2nd Edition). This is a great (and small) book that does an excellent job of teaching the language. To quote the authors (who also invented the language):

> "C is not a big language, and it is not well served by a big book."

Chapter 5 is about pointers and arrays. It explains this all quite well. Feel free to take some time to read the book.

> The following sizeof expressions always evaluate to 1:

Yes, that is generally true, but it is not guaranteed in the language spec so it is a good practice to use sizeof(char) to be safe.

> Maybe what we're looking for according to Stackoverflow is sizeof(char*)?

No, this would would be a buffer overflow **for sure**. char* is a pointer, which on our systems is 64-bits in size.

If sizeof(char) is 1 (8 bits!) using sizeof(char* in memset() will overwrite memory outside of our valid allocated area. Because malloc() (from the C runtime or "CRT") was used, the memory allocated came from the C language "heap" which is a large block of memory that gets suballocated as requested (e.g. by malloc()). This means it is not guaranteed to overflow the allocated operating system memory region (which would cause an abend like a trap E/protection fault/segmentation fault on Intel x86_64/AMD64 architecture) but it might. We don't know what exactly would happen but we do know that we would overwrite (corrupt!) a bunch of memory that we don't want to touch.

#52 - 02/02/2023 04:18 AM - Galya B

- File 5701a-r14487.diff added

Thank you for the explanation. Actually I was able to figure out the first part by reading the documentation for the functions. It's relatively easy to learn by examples. I struggled with the part of malloc allocating memory differently for pointer to pointer and for pointer only and how that is expressed.

A new commit r14487 fixes the len / size. Here is the diff. The other comments were addressed in r14486.

#53 - 02/02/2023 12:05 PM - Greg Shah

Code Review Task Branch 5701a Revisions 14485 and 14486

The changes are good.

#54 - 02/02/2023 12:05 PM - Greg Shah

At this point, we need to pause because of the freeze on trunk. We will rebase and merge when the "thaw" occurs.

#55 - 02/22/2023 08:10 AM - Galya B

Greg, I merged the changes to an up-to-date branch 5701b and there were no conflicts (r14485). When do I merge?

#56 - 02/22/2023 08:27 AM - Greg Shah

It won't be until some time next week. We have many performance changes and some regression fixes that need to go first.

#57 - 02/27/2023 08:40 AM - Greg Shah

- Related to Feature #7147: Make FWD log outputs consistent added

#58 - 03/14/2023 06:24 PM - Greg Shah

Please rebase 5701b from trunk rev 14505. Then if it passes your basic tests, you can merge to trunk.

#59 - 03/16/2023 04:58 AM - Galya B

5701b is merged to trunk under r14506 ~~and works fine, but there are regression issues with the previous revision r14505. Who was responsible for testing r14505?~~
Fix for the broken redirect condition in r14507.

#60 - 03/16/2023 11:42 AM - Galya B

- Status changed from Test to Feedback

#61 - 03/16/2023 11:46 AM - Greg Shah

- Status changed from Feedback to Closed

Feedback status is for when the task is blocked needing information from the Author and/or a customer. Leave it in Test when it has been merged. You can also post and ask for it to be closed.

#62 - 03/28/2023 10:26 AM - Constantin Asofiei

Galya, there is a regression in trunk revs 14506/14507.

This can be seen with appservers, where there is a MESSAGE "Message" VIEW-AS ALERT-BOX INFO BUTTONS OK. statement in an appserver call.

Can you point me where this issue may be?

#63 - 03/28/2023 10:29 AM - Galya B

Constantin Asofiei wrote:

> This can be seen with appservers, where there is a MESSAGE "Message" VIEW-AS ALERT-BOX INFO BUTTONS OK. statement in an appserver call.

What is the unexpected behavior? The only change in #5701 is redirecting System.out for appserver / scheduled batch processes.

#64 - 03/28/2023 10:33 AM - Constantin Asofiei

Galya Bogdanova wrote:

> Constantin Asofiei wrote:
>
> > This can be seen with appservers, where there is a MESSAGE "Message" VIEW-AS ALERT-BOX INFO BUTTONS OK. statement in an appserver call.

What is the unexpected behavior? The only change in #5701 is redirecting System.out for appserver / scheduled batch processes.

The FWD client dead-locks because FileChecker.isStdoutRedirected returns false. I see some possible issues:

- in ProcessClientBuilder.getSpawnArguments there is this code:

```
      if (outputToFile != null)
      {
         cmd.add("-O");
         cmd.add("server:clientConfig:outputToFile=" + outputToFile);
      }
```

- later on, in spawn.c around line 1300 there is this code:

```
      for (i = MIN_ARGS_NO_PASSWORD; i < argc; i++)
      {
         /* -O <outputToFile> */
         if ((strcmp("-O", argv[i]) == 0) && (++i < argc))
         {
            strcpy(output_file_name, argv[i]);
            output_to_file_opt_found = 1;
         }
      }
```

  Here, output_file_name gets assigned to server:clientConfig:outputToFile=somefile.log. There is no check to remove the '=' from the argument. I'm not sure if this is expected or not.
- this, in turn, uses it in launchP2JClient, this code:

```
   if (output_to_file_opt_found == 1)
   {
      int size = (strlen(output_file_name) + 1) * sizeof(char);
      arguments[length + 1] = (char*) malloc(size);
      memset(arguments[length + 1], '\0', size);
      strcpy(arguments[length + 1], output_file_name);
   }
```

  which just passes the 'bootstrap config and filename' to the arguments for the JVM launch. I think this is OK.

But, in filesys.c, there is this code:

```
JNIEXPORT jboolean JNICALL
   Java_com_goldencode_p2j_util_FileChecker_isStdoutRedirected(JNIEnv *env, jclass  jc)
{
   struct stat st;

   return (fstat(STDOUT_FILENO, &st) == 0) && // Means success call to fstat
          (S_ISCHR(st.st_mode) == 0) &&       // The output is not character device
          (S_ISREG(st.st_mode) != 0);         // but regular file
}
```

which will not see STDOUT as redirected.

#65 - 03/28/2023 10:37 AM - Constantin Asofiei

This patch seems to solve the issue:

```
--- old/src/com/goldencode/p2j/util/FileChecker.java
+++ new/src/com/goldencode/p2j/util/FileChecker.java
@@ -75,6 +75,7 @@

 import java.io.File;

+import com.goldencode.p2j.ui.client.chui.driver.batch.BatchHelper;
 import com.goldencode.util.*;

 /**
@@ -291,6 +292,6 @@
           consoleRedirected = Boolean.valueOf(isStdoutRedirected());
        }

-      return consoleRedirected.booleanValue();
+      return consoleRedirected.booleanValue() || BatchHelper.IS_OUTPUT_REDIRECTED.get();
    }
 }
```

But I don't know if this is the correct fix.

#66 - 03/28/2023 10:37 AM - Galya B

Constantin, this is troublesome. With #5701 stdout is not redirected in the native code, but programmatically in Java and FileChecker#isConsoleRedirected does not indicate that. I actually wasn't aware of that native method while depeloping, so I'll have to make amends now.

#67 - 03/28/2023 10:39 AM - Galya B

Constantin Asofiei wrote:

> This patch seems to solve the issue:
> [...]
>
> But I don't know if this is the correct fix.

Yes, this is what should be done.

#68 - 03/28/2023 10:41 AM - Galya B

Even better if IS_OUTPUT_REDIRECTED is moved to FileChecker.

#69 - 03/28/2023 10:42 AM - Constantin Asofiei

- Status changed from Closed to WIP

- % Done changed from 100 to 80

Galya Bogdanova wrote:

> Even better if IS_OUTPUT_REDIRECTED is moved to FileChecker.

Please prepare these changes in 5701c, and check if the filesys.c native code can be actually removed, if is no longer needed.

#70 - 03/28/2023 10:47 AM - Galya B

- % Done changed from 80 to 0

- Assignee deleted (Galya B)

- Priority changed from Normal to Low

- billable changed from No to Yes

Constantin Asofiei wrote:

> Please prepare these changes in 5701c, and check if the filesys.c native code can be actually removed, if is no longer needed.

It will be 5701d, the previous one was for the previous IS_OUTPUT_REDIRECTED patch. On it...

#71 - 03/28/2023 10:57 AM - Greg Shah

The 4GL knows when its STDIO is redirected and it changes its behavior based on that. I think we implemented the Java_com_goldencode_p2j_util_FileChecker_isStdoutRedirected() to duplicate that behavior.

Eugenie: Am I remembering correctly?

If so, we should not remove it.

#72 - 03/28/2023 11:07 AM - Eugenie Lyzenko

Greg Shah wrote:

> The 4GL knows when its STDIO is redirected and it changes its behavior based on that. I think we implemented the Java_com_goldencode_p2j_util_FileChecker_isStdoutRedirected() to duplicate that behavior.
>
> Eugenie: Am I remembering correctly?

Yes.

> If so, we should not remove it.

Yes, we need this call to be in native code.

#73 - 03/28/2023 11:08 AM - Constantin Asofiei

Greg, I think you are right. This may affect the redirected terminal in both ChUI and batch processes.

For batch processes, IIRC, STDOUT is the default destination for all UI statements. And OUTPUT TO can redirect it to a file (and this applies to ChUI/GUI, too).

There is at least this code in BatchPrimitives which I think needs to now real STDOUT redirection, and not the fact that 'output has been redirected to a file (or not)':

```
   @Override
   public void updateScreen(ScreenData screen)
   {
      if (isActive)
      {
         // Special processing for redirected STDOUT and Windows background mode
         if (FileChecker.isConsoleRedirected() ||
             (PlatformHelper.isUnderWindowsFamily() && OutputManager.isBatchInBackground()))
         {
```

We may need to choose carefully where the IS_OUTPUT_REDIRECTED flag is used, not just add it to FileChecker.isConsoleRedirected().

Galya: please do some tests in batch/appserver/ChUI, with code like this:

```
OUTPUT TO somefile.txt.
message "something".
output close.
```

This is the kind of STDOUT redirection I refer above.

#74 - 03/28/2023 11:42 AM - Galya B

- File 5701d-r14520.diff added

Constantin Asofiei wrote:

> Please prepare these changes in 5701c, and check if the filesys.c native code can be actually removed, if is no longer needed.

The native method isStdoutRedirected should still be in use for other cases like standalone batch where the redirect of stdout is done through the > syntax in the run command.

So now just adding this new condition to FileChecker#isConsoleRedirected.

Branch 5701d, r14520. Diff attached here.

#75 - 03/28/2023 11:45 AM - Galya B

I didn't receive notification for the notes from my last comment, so let me get up to date...

#76 - 03/28/2023 11:48 AM - Galya B

```
Updated by Galya Bogdanova about 1 hour ago

    billable changed from No to Yes
    Assignee deleted (Galya Bogdanova)
    Priority changed from Normal to Low
    % Done changed from 80 to 0
```

Lol. My... Is this the overwrite message warning for... I haven't set this myself.

#77 - 03/28/2023 12:03 PM - Greg Shah

- billable changed from Yes to No

- Assignee set to Galya B

- Priority changed from Low to Normal

- % Done changed from 0 to 80

#78 - 03/29/2023 02:19 AM - Constantin Asofiei

Galya Bogdanova wrote:

> Lol. My... Is this the overwrite message warning for... I haven't set this myself.

If someone else changes the status/priority/assignee at the task, and another person submits something without refreshing the page, Redmine gets 'confused' and messes up these.

#79 - 03/29/2023 05:46 AM - Galya B

Constantin Asofiei wrote:

> Galya: please do some tests in batch/appserver/ChUI, with code like this:
> [...]

If I understand right, to test the appserver I need to run a batch client for example that connects to appserver and runs the output to code on it.

I thought I knew how to do this, but any connect args I try, don't work with the new testcases, for example:

```
create server h.
h:connect("-H localhost -S 3333 -DirectConnect -AppService app_server").
```

```
h:connect("-AppService app_server -sessionModel Session-free").
```

```
h:connect("-H localhost -S fwd1 -sessionModel Session-free").
```

It's easy with the NS in 4GL. Here I get lost how to connect to the appserver. Maybe I'm missing configs.

#80 - 03/29/2023 06:06 AM - Galya B

Otherwise chui (terminal / swing) and batch work fine - generate the file and output any other display text.

#81 - 03/29/2023 08:36 AM - Constantin Asofiei

Galya, try this for appserver connect -AppService app_server -S 21200 -H localhost -sessionModel Session-free -DirectConnect.

Did you test the batch client with a program using OUTPUT TO somefile.txt?

#82 - 03/29/2023 08:56 AM - Galya B

Constantin Asofiei wrote:

> Did you test the batch client with a program using OUTPUT TO somefile.txt?

Yes. I've just tested appserver as well. The procedure I was running is doing output to, output stream to, showing alert-box and throwing AppError. The 2 files are created with all drivers. The two messages are recorded in outputToFile (for appserver), or to > (for batch), or displayed in chui terminal and swing apps.

```
using Progress.Lang.AppError.
using Progress.Lang.Object.

define variable oErr as AppError no-undo.
define stream stostream.

output stream stostream to value( "output.stream" ).

put stream stostream unformatted
"hello"
skip.

output stream stostream close.

output to output.to.
message "hello".
output close.

message "ERROR" view-as alert-box error.

oErr = new AppError('Error message from AppError constructor', 1) no-error.
oErr:AddMessage('Error message from AppError AddMessage', 2) no-error.
undo, throw oErr.
```

#83 - 03/29/2023 09:00 AM - Galya B

I've found a weird behavior just now. If the output to file is already existing, the text goes to outputToFile in appserver run. Let me verify if that was the original behavior.

#84 - 03/29/2023 09:10 AM - Galya B

Ok, that's straight weird. Every third run of the same procedure (without deleting the output to files) writes the file texts to outputToFile. I'm testing on the very old r14483. So the conclusion is this odd behavior is not related to #5701.

```
[?1049h[22;0;0t[1;24r(B[m[4l[?7h[?1h=ERROR
Error message from AppError constructor
ERROR
Error message from AppError constructor
hello
hello
ERROR
Error message from AppError constructor
ERROR
Error message from AppError constructor
hello
hello
ERROR
Error message from AppError constructor
ERROR
Error message from AppError constructor
hello
hello
ERROR
Error message from AppError constructor
```

#85 - 03/29/2023 09:12 AM - Galya B

Constantin, I think it is good to go... for 3rd time. 3rd time should be the lucky one.

#86 - 03/30/2023 02:18 AM - Constantin Asofiei

Galya, the changes in 5701d look OK.

Regarding your issue: I've checked your test and the hello you see in the appserver log is from the MESSAGE statement; this is correct. I've ran these tests in OE:

- runappsrv.p:

```
def var hs as handle.

create server hs.
message hs:connect("-AppService catest -sessionModel Session-free").
```

```
run msgtest.p on server hs (input "this is a test0").
run msgtest2.p on server hs (input "this is a test1").
run msgtest2.p on server hs (input "this is a test2").
run msgtest2.p on server hs (input "this is a test3").
message hs:disconnect().
```

- msgtest.p:

```
def input param ch as char.

MESSAGE "Message:" ch VIEW-AS ALERT-BOX INFO BUTTONS OK.
```

- msgtest2.p:

```
using Progress.Lang.AppError.
using Progress.Lang.Object.

def input param ch as char.

define variable oErr as AppError no-undo.
define stream stostream.

output stream stostream to value( "b.txt" ) append.

put stream stostream unformatted
"foo" ch
skip.

output stream stostream close.

output to a.txt append.
message "bar" ch.
output close.

message "ERROR" ch view-as alert-box error.

oErr = new AppError('Error message from AppError constructor ' + ch, 1) no-error.
oErr:AddMessage('Error message from AppError AddMessage ' + ch, 2) no-error.
undo, throw oErr.
```

which shows these logs:

```
[23/03/30@09:12:25.849+0300] P-004640 T-004644 1 AS -- (Procedure: 'msgtest.p' Line:3) Message: this is a test
0
[23/03/30@09:12:25.857+0300] P-004588 T-004592 1 AS -- (Procedure: 'msgtest2.p' Line:18) bar this is a test1
[23/03/30@09:12:25.858+0300] P-004588 T-004592 1 AS -- (Procedure: 'msgtest2.p' Line:21) ERROR this is a test1
[23/03/30@09:12:25.858+0300] P-004588 T-004592 1 AS -- (Procedure: 'msgtest2.p' Line:25) Error message from Ap
pError constructor this is a test1
[23/03/30@09:12:25.858+0300] P-004588 T-004592 1 AS -- (Procedure: 'msgtest2.p' Line:25) Error message from Ap
pError AddMessage this is a test1
[23/03/30@09:12:25.859+0300] P-004608 T-004612 1 AS -- (Procedure: 'msgtest2.p' Line:18) bar this is a test2
[23/03/30@09:12:25.859+0300] P-004608 T-004612 1 AS -- (Procedure: 'msgtest2.p' Line:21) ERROR this is a test2
[23/03/30@09:12:25.859+0300] P-004608 T-004612 1 AS -- (Procedure: 'msgtest2.p' Line:25) Error message from Ap
pError constructor this is a test2
[23/03/30@09:12:25.859+0300] P-004608 T-004612 1 AS -- (Procedure: 'msgtest2.p' Line:25) Error message from Ap
pError AddMessage this is a test2
[23/03/30@09:12:25.861+0300] P-004640 T-004644 1 AS -- (Procedure: 'msgtest2.p' Line:18) bar this is a test3
[23/03/30@09:12:25.861+0300] P-004640 T-004644 1 AS -- (Procedure: 'msgtest2.p' Line:21) ERROR this is a test3
[23/03/30@09:12:25.861+0300] P-004640 T-004644 1 AS -- (Procedure: 'msgtest2.p' Line:25) Error message from Ap
pError constructor this is a test3
[23/03/30@09:12:25.861+0300] P-004640 T-004644 1 AS -- (Procedure: 'msgtest2.p' Line:25) Error message from Ap
pError AddMessage this is a test3
```

but FWD shows this:

```
Message:this is a test0
ERRORthis is a test1
Error message from AppError constructor this is a test1
Error message from AppError constructor this is a test1
Error message from AppError AddMessage this is a test1
ERRORthis is a test2
Error message from AppError constructor this is a test2
Error message from AppError constructor this is a test2
Error message from AppError AddMessage this is a test2
bar this is a test3
bar this is a test3
ERRORthis is a test3
Error message from AppError constructor this is a test3
Error message from AppError constructor this is a test3
Error message from AppError AddMessage this is a test3
```

Keep in mind this is with the 5701d fix and FWD trunk. So FWD trunk loses some MESSAGE statements when writing them to log. Can these already be fixed by your other log-related tasks? Otherwise, we need to open a bug for this.

#87 - 03/30/2023 02:35 AM - Galya B

Constantin Asofiei wrote:

> Keep in mind this is with the 5701d fix and FWD trunk. So FWD trunk loses some MESSAGE statements when writing them to log. Can these already be fixed by your other log-related tasks? Otherwise, we need to open a bug for this.

I would say let's open the bug and block it until all log related tasks are merged, because it's confusing for me as well at this stage. Let's test it again after it's all done.

#88 - 03/30/2023 02:37 AM - Galya B

I think I did something to the repeated constructor msg, but atm not 100% sure.

#89 - 03/30/2023 02:43 AM - Galya B

I added some missing messages to the clientlog for #5753.

#90 - 03/30/2023 02:56 AM - Constantin Asofiei

Galya Bogdanova wrote:

> Constantin Asofiei wrote:
>
> > Keep in mind this is with the 5701d fix and FWD trunk. So FWD trunk loses some MESSAGE statements when writing them to log. Can these already be fixed by your other log-related tasks? Otherwise, we need to open a bug for this.
>
> I would say let's open the bug and block it until all log related tasks are merged, because it's confusing for me as well at this stage. Let's test it again after it's all done.

I agree, create a bug in Base Language and link it as related task to #5701 and other logging tasks.

#91 - 03/30/2023 04:12 AM - Galya B

- Related to Bug #7236: Inconsistencies in stderr and logs related to MESSAGE and AppError constructor added

#92 - 03/30/2023 04:14 AM - Galya B

- % Done changed from 80 to 100

- Status changed from WIP to Review

#93 - 03/30/2023 04:45 AM - Galya B

I was able to reproduce the issue with the hanging appserver and it's fixed in 5701d as I've mentioned earlier. Do I merge?

#94 - 03/30/2023 06:57 AM - Constantin Asofiei

Galya Bogdanova wrote:

> I was able to reproduce the issue with the hanging appserver and it's fixed in 5701d as I've mentioned earlier. Do I merge?

Yes, rebase 5701d from trunk rev 14520 and merge it.

#95 - 03/30/2023 07:13 AM - Galya B

Merged to trunk as r14521. It also fixes the repeating of log messages mentioned in #5753#note-79.

Do I have to write an email for the fix?

#96 - 03/30/2023 07:23 AM - Constantin Asofiei

Galya Bogdanova wrote:

> Merged to trunk as r14521. It also fixes the repeating of log messages mentioned in #5753#note-79.
>
> Do I have to write an email for the fix?

Yes, when a branch is merged to trunk, an email notification must be sent to the entire team. Also, make sure to archive the branch.

#97 - 04/06/2023 07:45 AM - Greg Shah

- Status changed from Review to Closed

Files

PID-Test.zip	5.09 KB	12/01/2022	Galya B
PID-Test-1.zip	5.23 KB	12/05/2022	Galya B
spawner-processes-unix-mode0.svg	14.6 KB	12/05/2022	Galya B
spawner-processes-win.png	48.7 KB	12/05/2022	Galya B
outputToFile-in-java.diff	2.22 KB	12/15/2022	Galya B
pid-in-outputToFile-fix.diff	5.22 KB	01/26/2023	Galya B
patch-pid-in-outputToFile-plus-win-fix.diff	10.4 KB	01/27/2023	Galya B
5701a-r14485.diff	13.1 KB	01/31/2023	Galya B
5701a-r14485.diff-2.png	244 KB	02/01/2023	Galya B
5701a-r14485.diff-1.png	246 KB	02/01/2023	Galya B
5701a-r14485.diff-3.png	244 KB	02/01/2023	Galya B
5701a-r14485.diff-4.png	210 KB	02/01/2023	Galya B
5701a-r14485.diff-5.png	259 KB	02/01/2023	Galya B
5701a-r14485.diff-6.png	227 KB	02/01/2023	Galya B
5701a-r14485.diff-7.png	145 KB	02/01/2023	Galya B
5701a-r14487.diff	1.17 KB	02/02/2023	Galya B
5701d-r14520.diff	5.24 KB	03/28/2023	Galya B