Base Language - Feature #4392

add -pf support for appserver CONNECT() method

11/08/2019 03:56 PM - Greg Shah

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

Description

Related issues:

Related to Database - Feature #3813: misc DB features part deux	**Closed**
Related to Runtime Infrastructure - Feature #7675: add missing runtime suppor...	**New**

History

#1 - 11/08/2019 03:57 PM - Greg Shah

Example:

h-appsrv:connect("-pf " + some-filename) no-error.

#2 - 01/24/2020 08:12 AM - Greg Shah

- Related to Feature #3813: misc DB features part deux added

#3 - 01/24/2020 08:12 AM - Greg Shah

- Status changed from New to WIP

- Assignee set to Igor Skornyakov

#4 - 04/24/2020 11:43 AM - Greg Shah

What is the status of this task?

#5 - 04/24/2020 11:46 AM - Greg Shah

- Related to deleted (Feature #3813: misc DB features part deux)

#6 - 04/24/2020 01:07 PM - Igor Skornyakov

Greg Shah wrote:

> What is the status of this task?

This was done in the scope of #3813.

#7 - 04/24/2020 01:21 PM - Greg Shah

This task is for the "server" handle-based resource (the appserver 4GL features, not database) and the method CONNECT() called on that handle. The implementation is in ServerImpl.connect(). It is not the same thing as the CONNECT language statement which is related to database connections and is implemented in ConnectionManager.connect().

As far as I know, this is not yet done.

FYI, we will want to use common code for as much of this as possible, since both cases use the same kind of command line/options string approach.

#8 - 04/24/2020 01:33 PM - Igor Skornyakov

Greg Shah wrote:

> This task is for the "server" handle-based resource (the appserver 4GL features, not database) and the method CONNECT() called on that handle. The implementation is in ServerImpl.connect(). It is not the same thing as the CONNECT language statement which is related to database connections and is implemented in ConnectionManager.connect().
>
> As far as I know, this is not yet done.
>
> FYI, we will want to use common code for as much of this as possible, since both cases use the same kind of command line/options string approach.

Oh, I see. Sorry. I do not think that it will be difficult.

#9 - 08/11/2023 10:49 AM - Greg Shah

- Related to Feature #3813: misc DB features part deux added

#10 - 08/11/2023 10:49 AM - Greg Shah

- Related to Feature #7675: add missing runtime support for 4GL command line parameters added

#11 - 12/11/2023 10:07 AM - Galya B

- Assignee changed from Igor Skornyakov to Galya B

- Status changed from WIP to Review

- % Done changed from 0 to 100

4392a r14865 ready for review.

Fixes the connect param "-pf" for servers. The code wasn't completed and was throwing the clients in infinite loop.

#12 - 12/11/2023 10:12 AM - Galya B

...and r14866 for the file header.

#13 - 12/11/2023 03:10 PM - Greg Shah

Code Review Task Branch 4392a Revisions 14865 and 14866

I did a quick review. The primary issue I see is that the code has switched to use local files. Since this code is run on the FWD server, using new File() is not a general solution. The 4GL compatible case requires reading the pf file from the FWD client.

#14 - 12/11/2023 03:10 PM - Greg Shah

Constantin: Please review.

#15 - 12/11/2023 03:56 PM - Constantin Asofiei

Some other notes, beside what Greg mentioned:

1. does -pf overwrite only the preceding options (the one which appear before -pf), and subsequent options (after the -pf) will overwrite the -pf values?
2. we need to double-check if recordOrShowError is really used (i.e. ERROR condition is not being thrown)
3. comments ('#') char are no longer being treated?

#16 - 12/12/2023 12:45 AM - Galya B

Greg Shah wrote:

> Code Review Task Branch 4392a Revisions 14865 and 14866
>
> I did a quick review. The primary issue I see is that the code has switched to use local files. Since this code is run on the FWD server, using new File() is not a general solution. The 4GL compatible case requires reading the pf file from the FWD client.

I see. I'll try to revert the approach then, but the original code was quite flawed. It was reading empty lines only, so it will take some time to figure it out.

Constantin Asofiei wrote:

> Some other notes, beside what Greg mentioned:
>
> 1. does -pf overwrite only the preceding options (the one which appear before -pf), and subsequent options (after the -pf) will overwrite the -pf values?

Yes, this is what I've tested in OE yesterday.

> we need to double-check if recordOrShowError is really used (i.e. ERROR condition is not being thrown)

I haven't introduced changes here, but I'll check the behavior.

> comments ('#') char are no longer being treated?

The behavior needs to be verified indeed.

#17 - 12/12/2023 12:47 AM - Galya B

Greg Shah wrote:

> Code Review Task Branch 4392a Revisions 14865 and 14866
>
> I did a quick review. The primary issue I see is that the code has switched to use local files. Since this code is run on the FWD server, using new File() is not a general solution. The 4GL compatible case requires reading the pf file from the FWD client.

By the way -pf for DB connect uses new File(). Is it acceptable there?

#18 - 12/12/2023 01:44 AM - Galya B

I've found the issue with the remote stream:

FileStream:

```
** OM  20210328          OE quirk: if last line is not EOL terminated, the readLn() method returns the
**                            previous read line (actually, if that EOL is missing the file is binary by
**                            definition).
```

But the pf file is usually expected to be one line and I managed to create it without end-of-line character, so I think it will be a common problem and we'll have to ask customers to add empty lines to make it work.

#19 - 12/12/2023 01:48 AM - Constantin Asofiei

Ovidiu, please see previous note - do you recall why that was added?

#20 - 12/12/2023 02:17 AM - Constantin Asofiei

Galya B wrote:

> By the way -pf for DB connect uses new File(). Is it acceptable there?

You mean the code in ConnectionManager$OptionsParser.parse? I don't think that is OK, it needs to use client-side streams, not server-side. Ovidiu: can this ever be called from the FWD server thread?

#21 - 12/12/2023 02:37 AM - Galya B

Constantin Asofiei wrote:

> Galya B wrote:
>
> > By the way -pf for DB connect uses new File(). Is it acceptable there?
>
> You mean the code in ConnectionManager$OptionsParser.parse?

Yes.

> we need to double-check if recordOrShowError is really used (i.e. ERROR condition is not being thrown)

The error doesn't stop the execution of the next statements in the procedure, so recordOrShowError seems correct.

r14867: reverted reading the pf file from remote file stream. Now working with the caveat in #4392-18.

#22 - 12/12/2023 09:09 AM - Greg Shah

> By the way -pf for DB connect uses new File(). Is it acceptable there?

No. That is not OK there either.

#23 - 12/12/2023 08:26 PM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> Ovidiu, please see previous note - do you recall why that was added?

I remember the moment I added this change. Most likely, I was 'polishing' the LOB support using xfer testcases project. I remember the behaviour was exactly as described, in H and commit of revision 12222: the next-to last line was appearing **twice** if the last line did not have the line terminator. Unfortunately, I did not remember the exact 4GL construct. I looked back in my archive but there are multiple issues fixed in this revision so I could not spot it.

#24 - 12/13/2023 01:39 AM - Galya B

I'll add a flag to handle both cases. By default it will be the old logic, since we can't pinpoint where it's used. And with the flag it will handle the standard files like -pf.

#25 - 12/13/2023 06:11 AM - Galya B

A note: Even with StreamFactory it's not always the client filesystem, but depends on the directory config server-side-resources/filesystem.

#26 - 12/13/2023 08:53 AM - Galya B

4392a r14868 Ready for review. Fixing the remote FileStream by adding a flag to determine if the file is supposed to have a duplicate last line.

#27 - 12/22/2023 12:16 PM - Constantin Asofiei

Review of 4392a rev 14868:

- ConnectHelper - is it possible to go into an infinite recursion in 4GL, if -pf is used to reference the same (or a previous, parsing unfinished) file? ConnectionManager's -pf parse code is aware of this.
- FileStream - **Ovidiu**: I understand the purpose of 'hasDuplicatedLastLine' flag, but assuming that this should be set only for LOB files, shouldn't this made by default 'false'? Did you ever see this behavior when reading from 'normal' input from streams?
- we need to fix the ConnectionManager part where it uses 'new File'

#28 - 12/22/2023 10:40 PM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> - FileStream - **Ovidiu**: I understand the purpose of 'hasDuplicatedLastLine' flag, but assuming that this should be set only for LOB files, shouldn't this made by default 'false'? Did you ever see this behavior when reading from 'normal' input from streams?

I am recalling this strange behaviour, but unfortunately, my memory has a hole regarding the exact testcase. Since mine is a edge-case (I tried to identify it these days, but I was not able any more :( ) the flag should definitely be set by default to false.

#29 - 01/01/2024 04:05 AM - Galya B

Constantin Asofiei wrote:

> - we need to fix the ConnectionManager part where it uses 'new File'

It sounds a related task, but in actual fact it's not. I need a test example.

#30 - 01/01/2024 04:09 AM - Galya B

Ovidiu Maxiniuc wrote:

> Constantin Asofiei wrote:
>
> - FileStream - **Ovidiu**: I understand the purpose of 'hasDuplicatedLastLine' flag, but assuming that this should be set only for LOB files, shouldn't this made by default 'false'? Did you ever see this behavior when reading from 'normal' input from streams?
>
> I am recalling this strange behaviour, but unfortunately, my memory has a hole regarding the exact testcase. Since mine is a edge-case (I tried to identify it these days, but I was not able any more :( ) the flag should definitely be set by default to false.

I would recommend to go with the safe default to true, as that is how it works for the moment. Although it might be a wrong behavior, it is the current one and noone has complained. I don't understand the implications of changing such a core method, so I can't do regression testing. If we decide to go with the "change it and see if someone screams", it should be a conscious decision.

#31 - 01/01/2024 09:09 AM - Galya B

Constantin Asofiei wrote:

> Review of 4392a rev 14868:
>
> - ConnectHelper - is it possible to go into an infinite recursion in 4GL, if -pf is used to reference the same (or a previous, parsing unfinished) file? ConnectionManager's -pf parse code is aware of this.

Since we're not up for reproducing the Stack Overflow in OE, the change is in r14869.

#32 - 01/08/2024 04:08 PM - Greg Shah

- we need to fix the ConnectionManager part where it uses 'new File'

> It sounds a related task, but in actual fact it's not. I need a test example.

Isn't this something like CONNECT my_db_name -pf some/path/to/pf_file.pf.? The .pf file would hold any options. I think it can even hold the database name if you preface the option with -db <dbname>.

If the path to the pf file can be found the client and not on the server, then the code will fail.

#33 - 01/10/2024 10:40 AM - Galya B

r14870 & 14871: Resolving DB CONNECT -pf file from the configured file system.

#34 - 01/10/2024 10:42 AM - Galya B

Ready for second review.

Note that #4392-30 was left unanswered and is not addressed.

#35 - 01/10/2024 10:45 AM - Greg Shah

> Note that #4392-30 was left unanswered and is not addressed.

Ovidiu: This is for you, I think.

#36 - 01/10/2024 10:48 AM - Galya B

Greg Shah wrote:

> > Note that #4392-30 was left unanswered and is not addressed.
>
> Ovidiu: This is for you, I think.

The summary: Constantin wanted to make the new flag false by default. Ovidiu couldn't remember the details of the original implementation and test cases. I decided to leave it with the original behavior (the flag to true) to prevent unforeseen issues. Best outcome is someone to summarize the tests related to the issue. I wasn't able to reason about the affected code parts, so my decision sounds reasonable.

#37 - 01/10/2024 10:53 AM - Greg Shah

Ovidiu: What is the list of testcases Galya can check?

#38 - 01/10/2024 03:49 PM - Ovidiu Maxiniuc

Greg Shah wrote:

> Note that #4392-30 was left unanswered and is not addressed.
>
> Ovidiu: This is for you, I think.

Sorry, I did not see the question in #4392-30.

You are right, Galya, the code generally works with the quirk in place and no one reported it as an issue. Until now. The problem here is that this is an edge-case and statistically, very few users reached this place (by using an invalid text file). This is the reason I still stand with my opinion from #4392-28 to be more lenient and, by default, to return the invalid line. In the future, if somebody reports a problem here we will have the chance to do more adjustments based on the new input. This is the downside of working with a black-box.

Greg Shah wrote:

> Ovidiu: What is the list of testcases Galya can check?

To check the quirk, I think the sftp://om@xfer.goldencode.com/opt/testcases/copy_lob tests should be run since this is the place I remember finding the original issue. Especially those which copy from a file. However, I looked over the project and the structure and input files are a bit changed since I encountered the problem. I do not think we have other tests which intensively works with (input) text streams, or I could not find them on that nor into the old testcases repo.
I guess navigation on a couple of current customer larger projects will create a chance for the execution to reach this spot.

#39 - 01/11/2024 03:37 AM - Galya B

The tests in copy_lob give the same results in both cases. I changed the flag hasDuplicatedLastLine to false everywhere.

r14873 ready for review.

#40 - 01/11/2024 12:05 PM - Greg Shah

Constantin/Ovidiu: Please review.

#41 - 01/12/2024 12:00 AM - Ovidiu Maxiniuc

Review of 4392a, 14869-14873 (from the previous review):

- ConnectionManager.java:
  - I like you cleaned up the import list. The IDEs usually collapse this section, but adds unwanted libraries when a wrong identifier is typed;
  - the iter.next(); at line 4926 may throw NoSuchElementException as it is not guarded by a hasNext();
- ServerConnectHelper.java
  - beside import list cleanup you use the bzr move/rename feature. Thumb up!

#42 - 01/12/2024 04:08 AM - Galya B

Ovidiu Maxiniuc wrote:

- ConnectionManager.java:
  - the iter.next(); at line 4926 may throw NoSuchElementException as it is not guarded by a hasNext();

Good catch, I tried to be lenient and leave some of the original code, but this seems to be a mistake. Now that I review wasBadOption again, it has other flaws as well, like not taking into consideration the case where the bad parameter is a flag without value. Reworked in r14874 (got rid of iter.next()).

- ServerConnectHelper.java
  - beside import list cleanup you use the bzr move/rename feature. Thumb up!

I've been brainwashed for the last year, doing manually move for the VCS :(

#43 - 01/19/2024 10:40 AM - Greg Shah

Ovidiu: Please do a final review of the latest changes (14874).

#44 - 01/22/2024 04:43 PM - Ovidiu Maxiniuc

- Status changed from Review to Internal Test

I did a review of all changes since 14864 (since it was branched from trunk). I found no severe issues, more like code style issues:

- ConnectionManager.java:
  - line 685 and next: the values of @param, @return, and @throws are not TAB-aligned. An empty line before the latter is missing;
  - line 4547: I know this is not your update, but please 'chop' the line down to 110 characters limit;
  - line 4822: small typo "CONNNECT"
  - line 4856: seeing the multiple question marks gives me impression that something is still unfinished.
  - line 4945: more like a personal preference: if the argument of ConnectOptionValue.option() is extracted into a local variable, the code can be written more compact an readable, on 2 lines only instead of 5;
- ParamFileReader.java:
  - lines 2-61: both headers (history and copyright) are indented by a space, probably caused by copy/paste in IDE;
  - line 3 and 74: The class javadoc is very short and actually misleading. I know the class exposes only two methods, but please add a proper description for it;
  - lines 147 and 219: the code style requires a {/} block even if there is only one statement in if / for / while;
- FileStream.java, StreamFactory.java, StreamDaemon.java & StreamBuilder.java
  - the copyright year needs to be updated (the H date should remain unchanged)

Then, please rebase to latest trunk. Hoping there are no issues during the process, the branch can be merged to trunk.

#45 - 01/23/2024 08:23 AM - Galya B

Ovidiu Maxiniuc wrote:

> - ConnectionManager.java:
>   - line 685 and next: the values of @param, @return, and @throws are not TAB-aligned. An empty line before the latter is missing;
>   - line 4547: I know this is not your update, but please 'chop' the line down to 110 characters limit;
>   - line 4822: small typo "CONNNECT"

All of these are not my updates.

> - line 4856: seeing the multiple question marks gives me impression that something is still unfinished.

It is. The explanation of the original author.

> - line 4945: more like a personal preference: if the argument of ConnectOptionValue.option() is extracted into a local variable, the code can be written more compact an readable, on 2 lines only instead of 5;

Let's make them 4 lines. Because the option line is 113 chars (whitespaces are not pasted correctly in Redmine, check in IDE) and gets wrapped, but the ternary operator should be properly separated into 3 lines.

```
                ConnectOptionValue option = ConnectOptionValue.option(opt.isText() ?
                                                                      arg :
                                                                      Integer.parseInt(arg));
```

```
                options.add(new AbstractMap.SimpleEntry<>(opt,option));
```

- line 3 and 74: The class javadoc is very short and actually misleading. I know the class exposes only two methods, but please add a proper description for it;

What are you misled to think of ParamFileReader if not -pf file helper? When you have such subjective thoughts, please share more.

- lines 147 and 219: the code style requires a {/} block even if there is only one statement in if / for / while;

This is copy pasted from ServerConnectHelper, added by Greg in r10745. I've never written if / for / while with missing curly braces in my career. It's also stated in ServerConnectHelper header, but I'll add the info to ParamFileReader as well.

> Then, please rebase to latest trunk. Hoping there are no issues during the process, the branch can be merged to trunk.

Changes applied: Question marks in JavaDoc removed, 5 lines down to 4. File headers copyright to 2024. Space in ParamFileReader file header removed. The code styling issues are not introduced by my code and I will abstain from meddling with them in view of the global picture.

Branch 4392a rebased on trunk r14940.

#46 - 02/07/2024 01:29 PM - Greg Shah

- Status changed from Internal Test to Review

Ovidiu: Please review.

#47 - 02/15/2024 01:11 PM - Ovidiu Maxiniuc

Review of 4392a/r14952, based on trunk r14940.

Galya B wrote:

> Changes applied: Question marks in JavaDoc removed, 5 lines down to 4.

I was thinking more like:

```
                Object optValue = opt.isText() ? arg : Integer.parseInt(arg);
                options.add(new AbstractMap.SimpleEntry<>(opt, ConnectOptionValue.option(optValue)));
```

Anyway, this was just a hint. We are not optimizing the number of lines of code :). Instead, the readability prevails. The aim is to quickly read an understand the code and spent the time on its semantics. Personally, I find easier to read code horizontally.
BTW, there are a few unneeded import statements newly added in ConnectionManager;

> File headers copyright to 2024. Space in ParamFileReader file header removed.

OK.

> The code styling issues are not introduced by my code and I will abstain from meddling with them in view of the global picture.

Fair enough. In fact, I spotted pieces of code like these in the new files; bzr just dumped the diff to console and I was not able to compare to the original. This is why I insist on using bzr move [--auto] instead of bzr delete + bzr add when a file is moved/renamed.

You will have to do the rebase operation again. Hopefully, there will not be any code conflicts.

I think the branch can be merged. If you have not yet done it, please smoke-test the branch with customer projects. If there are regressions, they will reveal early in the process.

#48 - 02/16/2024 03:53 AM - Galya B

Ovidiu Maxiniuc wrote:

> I was thinking more like:
> [...]
> Anyway, this was just a hint. We are not optimizing the number of lines of code :). Instead, the readability prevails. The aim is to quickly read an understand the code and spent the time on its semantics. Personally, I find easier to read code horizontally.

It would be good if we're optimizing the number of lines in methods (that helps with readability of the class): for example I don't like the style of separating arguments on different lines. As for the proposed change, I've missed to see this opportunity, so I'm applying it.

> BTW, there are a few unneeded import statements newly added in ConnectionManager;

Fixing.

> > The code styling issues are not introduced by my code and I will abstain from meddling with them in view of the global picture.

> Fair enough. In fact, I spotted pieces of code like these in the new files; bzr just dumped the diff to console and I was not able to compare to the original. This is why I insist on using bzr move [--auto] instead of bzr delete + bzr add when a file is moved/renamed.

I understand. Git is supported by the IDEs and on moving / renaming files it takes care of the VCS operations. But I'm doing the manual move for quite some time already. It's just that in this case some methods were moved to other classes. I think the project needs a linter to take care of such issues and block merges instead of code styling be a a thing (or not be, depending on the reviewer) in PRs.

> You will have to do the rebase operation again. Hopefully, there will not be any code conflicts.

4392a rebased on trunk r14986. Both requested improvements applied in r14999.

> I think the branch can be merged. If you have not yet done it, please smoke-test the branch with customer projects. If there are regressions, they will reveal early in the process.

To be able to test -pf in a customer project, I need to know how it's used currently. I can find some references to -pf in a customer project, but none looks clear if it's used in the test env, so I'll need advice from someone who is familiar with the project.

The worst is that DB -pf can't be even tested in testcases, because the default database is already connected. I've tested only the fact that -db is read from -pf and the thrown exception includes the name of the db.

#49 - 02/16/2024 05:57 PM - Ovidiu Maxiniuc

The smoke test I was thinking was to regression test the branch, to have a minimum certainty that there are no bugs added to code. With regard to the newly implemented stuff, we need a project which already has property files used in legacy code but FWD was not processing it correctly, up to this moment.

#50 - 02/20/2024 03:06 AM - Galya B

I've started one of the customer projects with 4392a and it seems to be working.

Greg, it should be ready for merge.

#51 - 02/20/2024 09:33 AM - Greg Shah

You can merge to trunk after 7019d.

#52 - 02/20/2024 09:33 AM - Greg Shah

- Status changed from Review to Merge Pending

#53 - 02/20/2024 09:54 AM - Galya B

4392a was merged to trunk as rev. 14995 and archived.

#54 - 02/20/2024 09:56 AM - Greg Shah

- Status changed from Merge Pending to Test