# Base Language - Bug #4453

## Handling of filename with quotation mark broken in control_flow.rules

12/06/2019 02:33 PM - Roger Borrello

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

**Description**

### History

**#1 - 12/06/2019 02:38 PM - Roger Borrello**

# Description

A filename with a quotation mark (albeit a typo) is not processed properly.

## Testcase

run myfile.p".

You'd expect rules/convert/literals.rules to handle this, but it contains specific protections when the parent of FILENAME is KW_RUN:

```
      <!-- filenames emit as a string EXCEPT in the case of a RUN statement
           which is handled elsewhere -->
      <rule>type == prog.filename and parent.type != prog.kw_run
         <action>
            createPeerAst(java.string,
                          progressToJavaString(sprintf("'%s'", text)),
                          closestPeerId)
         </action>
      </rule>
```

Instead, it is left up to rules/convert/control_flow.rules which has specific checks for double-quote ("), single-quote ('), or tilde (~) inside the filename.

```
            <!-- emit the procedure's name -->
            <action>legacyname = null</action>
            <rule>childref.type == prog.filename
               <!-- This is required for additional progressToJavaString() call, it needs a
                    quotes but only when the filename does not have ",' or ~ inside
                    -->
               <rule>childref.text.indexOf('"') != -1 or
                     childref.text.indexOf("'") != -1 or
                     childref.text.indexOf('~') != -1
                  <action>legacyname = childref.text</action>
                  <action on="false">
                     legacyname = ecw.progressToJavaString(sprintf("'%s'", childref.text), true)
                  </action>
               </rule>
            </rule>
            <rule>childref.type == prog.int_proc
               <action>legacyname = childref.text</action>
            </rule>
            <rule>legacyname != null
               <action>
                  createJavaAst(java.string, legacyname, refid)
               </action>
```

```
            </rule>
```

I fail to see the case where you would handle the filename of the run statement special.

#### #2 - 12/06/2019 02:39 PM - Roger Borrello

Roger Borrello wrote:

> # Description
>
> A filename with a quotation mark (albeit a typo) is not processed properly.

From Constantin:

What I can tell you is that those characters (",',~) are valid in a filename. At some point, we converted a RUN FILENAME to a direct Java call, and not via ControlFlowOps - so this may be a reminiscent of that.

I haven't worked on (or don't recall) checking all possible chars which can appear as a filename literal for the RUN statement, so we may have something missing there. What I can agree is that a RUN a.p" 'strips' the " char. More, this seems to be related not only to the quote, but to any char: RUN a.p_lots_of_garbage_here works in 4GL (at least on Windows).

#### #3 - 12/06/2019 02:55 PM - Roger Borrello

Roger Borrello wrote:

> Roger Borrello wrote:
>
> > # Description
> >
> > A filename with a quotation mark (albeit a typo) is not processed properly.
>
> From Constantin:
>
> What I can tell you is that those characters (",',~) are valid in a filename. At some point, we converted a RUN FILENAME to a direct Java call, and not via ControlFlowOps - so this may be a reminiscent of that.
>
> I haven't worked on (or don't recall) checking all possible chars which can appear as a filename literal for the RUN statement, so we may have something missing there. What I can agree is that a RUN a.p" 'strips' the " char. More, this seems to be related not only to the quote, but to any char: RUN a.p_lots_of_garbage_here works in 4GL (at least on Windows).

I saw that Eugenie had updated control_flow.rules on 2nd of October, 2014:

```
** 070 EVL 20140210          Fixup the run path\to\\program.p statements for Windows path strings.
**                           In this case the single \ character is not a part of the escaped
**                           character so we need to process the filename by
**                           progressToJavaString() call.  Also the name of the file should
**                           additionally be quoted because the transformation removes the quotes
**                           while processing. However the additional transformation should be
**                           disabled when filename has one of the following characters inside:
**                           '(single quote), "(double quote) or ~(tilde).  Also in Windows we
**                           need to make difference between handling of the path containing
**                           strings and other ones.
```

If this is supposed to have been fixed via the updte:

```
10509       evl@gol |             <rule>childref.type == prog.filename
                    |                <!-- This is required for additional progressToJavaString() call, it need
s a
                    |                     quotes but only when the filename does not have ",' or ~ inside
                    |                     -->
                    |                <rule>childref.text.indexOf('"') != -1 or
                    |                      childref.text.indexOf("'") != -1 or
                    |                      childref.text.indexOf('~') != -1
                    |                   <action>legacyname = childref.text</action>
                    |                   <action on="false">
                    |                      legacyname = ecw.progressToJavaString(sprintf("'%s'", childref.text
), true)
                    |                   </action>
                    |                </rule>
```

Eugenie, can you provide any more details or examples? When a file name is runit.p" the Java should be "runit.p\"".

#### #5 - 12/06/2019 03:00 PM - Constantin Asofiei

Roger Borrello wrote:

> When a file name is runit.p" the Java should be "runit.p\"".

Argh, I don't understand, my testcase of run a.p_garbage is a very-very limited bug in 4GL. I can't make it work with other filenames, b.p_garbage, runit.p_garbage, x.p_garbage, these don't find the file without _garbage suffix. Only RUN a.p_garbage finds the a.p file.

Roger, can you do a quick check in 4GL and double-check this? Create a a.p file with a single message statement, and use RUN a.p_garbage to execute it.

#### #6 - 12/06/2019 03:03 PM - Constantin Asofiei

Constantin Asofiei wrote:

> Roger Borrello wrote:
>
> > When a file name is runit.p" the Java should be "runit.p\"".
>
> Argh, I don't understand, my testcase of run a.p_garbage is a very-very limited bug in 4GL. I can't make it work with other filenames, b.p_garbage, runit.p_garbage, x.p_garbage, these don't find the file without _garbage suffix. Only RUN a.p_garbage finds the a.p file.
>
> Roger, can you do a quick check in 4GL and double-check this? Create a a.p file with a single message statement, and use RUN a.p_garbage to execute it.

Nevermind, I had a a.r which has precedence (in the lookup) over a a.p_garbage filename.

#### #7 - 12/06/2019 03:35 PM - Constantin Asofiei

And the implication for my 'garbage suffix' misdirection: if the 4GL code (like RUN a.p_garbage) is tested only with a a.r file, then the RUN statement will work fine in 4GL, it will find the a.r program. This case is not treated in FWD; and this is not limited to the quote character.

So, if you leave in the original code runit.p", and this code was only tested with runit.r, then that IS valid 4GL code. It will run fine. If we fix FWD to add the quote to the filename, then FWD will not find it, because FWD has no knowledge (currently) that the target was supposed to be in a .r file!

#### #8 - 12/06/2019 04:15 PM - Roger Borrello

Constantin Asofiei wrote:

> And the implication for my 'garbage suffix' misdirection: if the 4GL code (like RUN a.p_garbage) is tested only with a a.r file, then the RUN statement will work fine in 4GL, it will find the a.r program. This case is not treated in FWD; and this is not limited to the quote character.
>
> So, if you leave in the original code runit.p", and this code was only tested with runit.r, then that IS valid 4GL code. It will run fine. If we fix FWD to add the quote to the filename, then FWD will not find it, because FWD has no knowledge (currently) that the target was supposed to be in a .r file!

We may have to shelve any implementation of that until we come across a scenario where this occurs. Greg can weigh in, if that's not the case.

I am having trouble understanding why we'd ever ***not*** use progressToJavaString on the program filename. The check for ",', or ~ looks wrong in any case. "If there isn't a " or there isn't a ' or there isn't a ~ then use filename as-is" is incorrect. It would only work if all 3 where in the filename.

```
<!-- This is required for additional progressToJavaString() call, it needs a
     quotes but only when the filename does not have "," or ~ inside
     -->
<rule>childref.text.indexOf('"') != -1 or
      childref.text.indexOf("'") != -1 or
      childref.text.indexOf('~') != -1
   <action>legacyname = childref.text</action>
   <action on="false">
      legacyname = ecw.progressToJavaString(sprintf("'%s'", childref.text), true)
   </action>
</rule>
```

Shouldn't that be changed to:

```
<!-- This is required for additional progressToJavaString() call, it needs a
     quotes but only when the filename does not have "," or ~ inside
     -->
<rule>childref.text.indexOf('"') == -1 and
      childref.text.indexOf("'") == -1 and
      childref.text.indexOf('~') == -1
   <action>legacyname = ecw.progressToJavaString(sprintf("'%s'", childref.text), true)</action>
   <action on="false">
      legacyname = childref.text
   </action>
</rule>
```

I will look for any testcases Eugenie checked in around that time. Or he could weigh in.

#### #9 - 12/06/2019 04:37 PM - Constantin Asofiei

Rev 10509 was part of #1634 - there's lots of discussion for tilde and quote chars, in paths.

#### #10 - 12/06/2019 05:01 PM - Roger Borrello

Constantin Asofiei wrote:

> Rev 10509 was part of #1634 - there's lots of discussion for tilde and quote chars, in paths.

Very interesting read. I found a testcase Eugenie attached. It looks like the below, and I included the proper generated output:

```
def stream outfile.
output stream outfile to "string_cvt.log".

put stream outfile "~\" skip.
put stream outfile "\\" skip.
/* MANY MORE TESTS CREATED BY EVL GO HERE */
put stream outfile "~/" skip.
put stream outfile "\\\\" skip.
put stream outfile "Alternative\path" skip.
put stream outfile "Alternative/path" skip.
put stream outfile 'path\to\''something' skip.
put stream outfile "path\to\""something" skip.
put stream outfile "path\to\~~tsomething" skip.

output stream outfile close.

/* Should result in:

\
\\
/
\\\\
Alternative\path
Alternative/path
path\to\'something
path\to\"something
path\to\~tsomething

*/
```

In our current 4207a, we get a parsing error:

```
--------------------------------------------------------------------------------
Scanning Progress Source (preprocessor, lexer, parser, persist ASTs)
--------------------------------------------------------------------------------

./string_cvt.p
Failure in file './string_cvt.p':
com.goldencode.ast.AstException: Error processing ./string_cvt.p
        at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:992)
        at com.goldencode.p2j.uast.ScanDriver.lambda$scan$0(ScanDriver.java:375)
        at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:410)
        at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:248)
        at com.goldencode.p2j.convert.TransformDriver.runScanDriver(TransformDriver.java:354)
        at com.goldencode.p2j.convert.TransformDriver.front(TransformDriver.java:225)
        at com.goldencode.p2j.convert.TransformDriver.executeJob(TransformDriver.java:864)
        at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:983)
Caused by: line 27:2: expecting ''', found '<EOF>'
        at com.goldencode.p2j.uast.ProgressLexer.nextToken(ProgressLexer.java:3303)
        at com.goldencode.p2j.uast.LexerDumpFilter.nextToken(LexerDumpFilter.java:201)
        at antlr.TokenStreamHiddenTokenFilter.consume(TokenStreamHiddenTokenFilter.java:38)
        at antlr.TokenStreamHiddenTokenFilter.nextToken(TokenStreamHiddenTokenFilter.java:134)
        at com.goldencode.ast.ManagedHiddenStreamTokenFilter.nextToken(ManagedHiddenStreamTokenFilter.java:99)
        at antlr.TokenBuffer.fill(TokenBuffer.java:69)
        at antlr.TokenBuffer.LT(TokenBuffer.java:86)
        at antlr.LLkParser.LT(LLkParser.java:56)
        at com.goldencode.p2j.uast.ProgressParser.mergeUntilWS(ProgressParser.java:2229)
```

```
        at com.goldencode.p2j.uast.ProgressParser.malformed_symbol(ProgressParser.java:10199)
        at com.goldencode.p2j.uast.ProgressParser.lvalue(ProgressParser.java:17410)
        at com.goldencode.p2j.uast.ProgressParser.put_stmt(ProgressParser.java:33846)
        at com.goldencode.p2j.uast.ProgressParser.stmt_list(ProgressParser.java:26668)
        at com.goldencode.p2j.uast.ProgressParser.statement(ProgressParser.java:8661)
        at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:7223)
        at com.goldencode.p2j.uast.ProgressParser.block(ProgressParser.java:6910)
        at com.goldencode.p2j.uast.ProgressParser.external_proc(ProgressParser.java:6837)
        at com.goldencode.p2j.uast.AstGenerator.parse(AstGenerator.java:1524)
        at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:987)
        ... 7 more
```

#### #11 - 12/06/2019 05:16 PM - Greg Shah

```
               <!-- This is required for additional progressToJavaString() call, it needs a
                    quotes but only when the filename does not have "," or ~ inside
                    -->
               <rule>childref.text.indexOf('"') == -1 and
                     childref.text.indexOf("'") == -1 and
                     childref.text.indexOf('~') == -1
                  <action>legacyname = ecw.progressToJavaString(sprintf("'%s'", childref.text), true)</action>
                  <action on="false">
                     legacyname = childref.text
                  </action>
               </rule>
```

I disagree with this.  If all three are -1 then the progressToJavaString() is NOT needed.  I think the entire point of using progressToJavaString() is to properly escape valid chars like the quote characters.  On the other hand, the use of tilde in a name should not be treated like an escape character which progressToJavaString() would do.  The code is wrong but it is not clear what is right.  More analysis is needed.

#### #12 - 12/06/2019 05:33 PM - Roger Borrello

Greg Shah wrote:

> [...]
>
> I disagree with this.  If all three are -1 then the progressToJavaString() is NOT needed.  I think the entire point of using progressToJavaString() is to properly escape valid chars like the quote characters.  On the other hand, the use of tilde in a name should not be treated like an escape character which progressToJavaString() would do.  The code is wrong but it is not clear what is right.  More analysis is needed.

If we exclude "" from the check in control_flow.rules, we get these results:

```
run runfile.p".
run runfile.p'.
run runfile.p~.
```

Results in:

```
        ControlFlowOps.invoke("runfile.p\"");
        ControlFlowOps.invoke("runfile.p'");
        ControlFlowOps.invoke("runfile.p");
```

Agreed... more analysis, and reading of #1634

#### #13 - 03/03/2020 02:28 PM - Roger Borrello

*- Status changed from New to WIP*

#### #14 - 03/03/2020 02:46 PM - Roger Borrello

Task branch 4207a was merged to trunk as revision 11344. There have not been any regressions or new issues related to this since customer code was updated. If this is more an issue of the .r file getting precedence, perhaps the title should be changed?