# Base Language - Feature #1648

## review all filesystem support (runtime) on Windows and fix/resolve any issues

10/21/2012 01:09 PM - Greg Shah

| | | | |
|---|---|---|---|
| **Status:** | Closed | **Start date:** | 11/20/2012 |
| **Priority:** | Normal | **Due date:** | |
| **Assignee:** | Eugenie Lyzenko | **% Done:** | 100% |
| **Category:** | | **Estimated time:** | 80.00 hours |
| **Target version:** | Runtime Support for Server Features | | |
| **billable:** | No | **vendor_id:** | GCD |

**Description**

**Related issues:**

| | | | |
|---|---|---|---|
| Related to Base Language - Bug #2130: fix propath assignment | **Closed** | **05/18/2013** | **05/20/2013** |

### History

**#1 - 10/31/2012 01:33 PM - Greg Shah**

*- Target version set to Milestone 12*

**#2 - 10/31/2012 01:34 PM - Greg Shah**

*- Target version changed from Milestone 12 to Milestone 7*

**#3 - 11/20/2012 02:24 PM - Greg Shah**

*- Start date set to 11/20/2012*

*- Assignee set to Eugenie Lyzenko*

*- Estimated time changed from 40.00 to 80.00*

*- Status changed from New to WIP*

Please review the streams.odt and file_system.odt chapters of the Conversion Reference. Both chapters have summary tables of the features that are supported. For each of the features in those summary tables, please find (or create) a testcase. Then please convert and run those testcases on P2J in both Linux and Windows platforms. Document any differences and investigate the root causes. If the fix is reasonable, you will be authorized to create the fix.

**#4 - 11/20/2012 02:26 PM - Eugenie Lyzenko**

OK.

**#5 - 11/23/2012 04:42 PM - Eugenie Lyzenko**

For now the following features described in streams.odt chapter have testcases to demonstrate conversion in Windows. All tests have been converted successfully and were executed in Windows environment.

| | |
|---|---|
| DEFINE STREAM | redirected_training11.p |
| EXPORT | redirected_training25.p |
| IMPORT | redirected_training29.p |
| INPUT CLOSE | stream_training5.p |
| INPUT FROM | stream_training6.p |
| INPUT THROUGH | stream_training11.p, stream_training3.p (only conversion works, no running native support in Windows) |
| OUTPUT CLOSE | stream_training4.p |
| OUTPUT TO | stream_training1.p, stream_training2.p |
| PAGE-SIZE | streamed_paged_frame1.p |
| PUT | redirected_training39.p |
| READKEY | stream_training5.p |
| TERMINAL | redirected_training9.p |

At this time there are no problem discovered with conversion/running the testcases in Windows. Continue investigation.

#### #6 - 11/26/2012 04:13 PM - Eugenie Lyzenko

Need to clarify in requirements for the following resources:

```
CLIPBOARD      - not supported yet
Device or File - in Linux the devices are files so no special samples required. In Windows there is no support
 for direct device access in 4GL and in P2J not supported as well.
OS-DIR         - not supported
PRINTER        - not supported
```

I guess we do not need the convertible/runnable sample test cases in Windows. Correct?

#### #7 - 11/26/2012 04:50 PM - Greg Shah

Correct.

#### #8 - 11/27/2012 06:06 PM - Eugenie Lyzenko

The second part of the stream chapter:

```
INPUT-OUTPUT CLOSE    input_output_thru_test.p
INPUT-OUTPUT THROUGH  input_output_thru_test.p (only conversion works, no running native support in Windows)
LINE-COUNTER          streamed_paged_frame2.p
OUTPUT THROUGH        output_thru_test.p (only conversion works, no running native support in Windows)
PAGE                  redirected_report3_7.p
PAGE-NUM(BER)         line_counter_test.p
SEEK Statement        closed_stream_error_29.p
SEEK()                seeking_file_reader.p
Child Process         caller.p + called_1.p + called_2.p running together
```

The only difference found at this time is related to the Windows line separator char. Trying to use input file with more than one line from Linux in Windows environment produce the error because P2J runtime in Windows+SET statement with INPUT FROM considers the file as one long line not making separation with 0x0A character. Converting the input files to Windows style fixes the problem.

Continue investigation with file_system.odt chapter.

#### #9 - 11/27/2012 06:58 PM - Greg Shah

It is an interesting finding (about the Linux line ending not being honored in P2J on Windows). Please test the same case in 4GL on Windows. In other words: does the 4GL properly honor the Linux line end the same as the Windows line end. If so, then please do work on a fix for P2J and propose it.

#### #10 - 11/27/2012 07:17 PM - Eugenie Lyzenko

> Please test the same case in 4GL on Windows. In other words: does the 4GL properly honor the Linux line end the same as the Windows line end. If so, then please do work on a fix for P2J and propose it.

Do you mean we need to test in true 4GL environment? And we have access to the 4GL in the customer's development system? Correct?

#### #11 - 11/28/2012 08:18 AM - Greg Shah

Yes and yes. Correct.

#### #12 - 11/28/2012 12:51 PM - Eugenie Lyzenko

The testing respective 4GL program in Windows OpenEdge10 discovers the 4GL accepts both Linux and Windows style line end and handles them the same way. There is no difference for Windows 4GL runtime between 0x0A and 0x0D 0x0A. So if in all Windows system this feature is the same - to duplicate this behavior we need a fix to make our P2J line-separator neutral.

#### #13 - 11/28/2012 01:27 PM - Greg Shah

OK. Please propose a fix as part of this task.

#### #14 - 11/29/2012 03:42 PM - Eugenie Lyzenko

The investigation results:
Affected code is located in com.goldencode.p2j.util package, classes Stream and FileStream.
The file Stream.java:

```
...
constant public static final String NEWLINE = System.getProperty("line.separator");
...
method public String readFieldWorker(BaseDataType var)
...
(1)                   sb.append(NEWLINE).append(more);
...
method protected String readLineCleanup(StringBuilder sb)
...
(2)       int idx = sb.indexOf(NEWLINE);
...
```

The file FileStream.java:

```
method public String readLn()
...
(3)       while (sb.indexOf(NEWLINE) < 0)
...
```

The all points 1-3 are the subject of modification. We have the right behavior if changing NEWLINE to "\n". Or simply substitute
NEWLINE = System.getProperty("line.separator");
with
NEWLINE = "\n";
The other related constant is(Stream class):
public static final int NEWLINE_LEN =
System.getProperty("line.separator").length();
The value of this constant does not affect the behavior in this case. Another word we may have NEWLINE_LEN == 1 or 2 - there is no difference.
So we can either change NEWLINE constant or change all 3 using cases above to leave remaining logic untouched. This is the simple fix. In general I would suggest to change the line separator to only hardcoded value "\n". This will work in both Linux and Windows environment. As far as I know the operating systems the 4GL is working is: Unix(different versions), Linux(different distros), MAC OS X and Windows. In these cases the char '\n' is the only one that exists in all of them(except some old MAC OS). So hardcoding this only value for P2J gives the ability to separate lines in both Windows and Linux. The other possible approach is to have the option in server.xml configuration file. Then P2J will check if there is the configured option in server.xml and if we redefine it - P2J engine will replace the value returned by System.getProperty("line.separator") call with one from server.xml config file.

Really it will be useful to know how exactly the native 4GL handles this situation. I'll try to dig some documentation. May be there are the Progress config file to set up such compatibility? Another word what if the input file will come from some exotic OS, like QNX, OS/390 or OS/400. How the Progress will handle them? It we would know this 4GL internal behavior - we can just simulate it and it will be the best compatibility level I guess.

#### #15 - 11/30/2012 05:10 PM - Greg Shah

Is case 1 really a problem?  That is about writing a newline, not searching for one.  I think it should write the local platform's newline (whatever it is).  What specific problem is being caused by that case?

In regards to cases 2 and 3, the answer is to match on all 3 of the following possible newline combinations:

\n  (0x0A - used in Unix/Linux)
\r\n (0D0A - used in Windows)
\r (0x0D - used on MacOS)

The key is matching any of them as a newline.  Anytime you see either a \r or \n, you have a newline BUT sometimes the \r may be followed by a \n and in that case you take the combination as the newline (it isn't 2 newlines).

I think if we implement this way, it will probably match how the 4GL does it.  It is possible the 4GL would fail on the MacOS case, so please check this.

#### #16 - 11/30/2012 06:01 PM - Eugenie Lyzenko

I just retested and have the correction. The case 1 can be left untouched. And moreover case 2 can be unchanged too. The remaining problem is the case 3 in FileStream.java where we walking through the lines reading the file. I did start changes with Stream class and changes there fix the issue only partially.

The fix in file FileStream.java overrides the Stream.java and eliminate mismatches there. So we really need only one change in FileStream class.

#### #17 - 11/30/2012 06:12 PM - Eugenie Lyzenko

All the following calls from the file_system.odt have testcases, converts and run in Windows environment:

```
CURRENT-LANGUAGE   file_system_training17.p
FILE-INFO          file_system_training29.p, file_system_training30.p, file_system_training31.p, file_system_t
raining32.p, file_system_training33.p, file_system_training34.p, file_system_training35.p, file_system_trainin
g36.p, file_system_training37.p
OPSYS              file_system_training0.p
OS-APPEND          file_system_training2.p
OS-COPY            file_system_training3.p, file_system_training4.p,
OS-CREATE-DIR      file_system_training5.p, file_system_training6.p, file_system_training7.p
OS-DRIVES          file_system_training18.p
OS-ERROR           file_system_training23.p
OS-DELETE          file_system_training8.p, file_system_training9.p, file_system_training23.p
OS-GETENV          file_system_training11.p
OS-RENAME          file_system_training12.p, file_system_training13.p, file_system_training14.p
```

For now the only interesting finding is:
Consider the path string for the file system calls like OS-CREATE-DIR containing Linux path separator ('/'). The P2J runtime does not transform the string like "One/Two" into Windows style. However the Windows handles the Linux path separator correctly in all respective calls above. So this is not an issue for P2J implementation in Windows environment. Continue working.

#### #18 - 12/03/2012 09:43 AM - Greg Shah

Although the Stream use of searching for NEWLINE isn't causing a problem right now, I still want it to be fixed.  Try to implement the solution using a helper method in Stream and then reuse the same helper method from both locations (cases 2 and 3).

#### #19 - 12/03/2012 10:38 AM - Eugenie Lyzenko

OK. The suggested helper is the method in Stream class:

```
   protected int indexOfNewLine(StringBuilder sb)
   {
      int result = -1;

      // Use system default first
```

```
      result = sb.indexOf(NEWLINE);
      // Default new line not found, try Linux style first
      if (result < 0)
         result = sb.indexOf("\n");
      // Then try Windows style
      if (result < 0)
         result = sb.indexOf("\r\n");
      // And finally try special MAC OS versions
      if (result < 0)
         result = sb.indexOf("\r");

      return result;
   }
```

Then change the 2 and 3 cases this way:

```
Stream.readLineCleanup
{
...from
      int idx = sb.indexOf(NEWLINE);
to
      int idx = indexOfNewLine(sb);
...
}

FileStream.readLn()
{
...from
      while (sb.indexOf(NEWLINE) < 0)
to
      while (indexOfNewLine(sb) < 0)
...
}
```

Tested in Windows and confirmed as working solution for Linux line terminator and for MAC OS too("\r").

And one more news: The MAC OS style line separator is not handled properly in native 4GL on Windows. Not the same way as P2J in Windows+Linux new line but not as expected for correct work.

#### #20 - 12/03/2012 02:50 PM - Greg Shah

It looks good.  Remove the MacOS support from indexOfNewLine() and go with it.

#### #21 - 12/03/2012 04:29 PM - Eugenie Lyzenko

Do we need this change to be integrated into current P2J build? I mean regression testing and checking in to bzr? Or we need to hold this change for now until all features of this task will be tested?

#### #22 - 12/04/2012 09:06 AM - Greg Shah

Hold it until we have all the changes ready for the task.

#### #23 - 12/04/2012 07:03 PM - Eugenie Lyzenko

The following calls has testcases to verify conversion result in Windows:

```
PROGRAM-NAME       file_system_training22.p and file_system_training21.p
PROGRESS           file_system_training19.p
PROMSGS            file_system_training24.p
PROPATH function   file_system_training15.p(problems while getting the default value)
PROPATH statement  file_system_training16.p not supported on conversion level but available for manual code
PROVERSION         file_system_training20.p
SEARCH             file_system_training28.p
SESSION            file_system_training25.p, file_system_training26.p, file_system_training27.p
```

There are two issues have been detected in converted code. One is knowing and related to the PROPATH statement conversion limitation which is described in file-system.odt document.

The other one is the difference in PROPATH function result. In Linux the result value is taken from the directory.xml file and in case there is no value defined in directory - the default value is ".:". This is hardcoded in EnvironmentOps.getSearchPath()

```
...
         propath = Utils.getDirectoryNodeString(null,
                                                "searchpath",
                                                ".:");
...
```

to return the default value of ".:". In Windows this behavior become incorrect. Looks like we need to insert the OS detection code into P2J here or as alternative solution we can add the following entry into server directory definition to some place where it can be found by EnvironmentOps.getSearchPath():

```
        <node class="container" name="runtime">
          <node class="container" name="default">
...
            <node class="string" name="searchpath">
              <node-attribute name="value" value=".;"/>
            </node>
...
```

#### #24 - 12/05/2012 02:45 PM - Greg Shah

```
One is knowing and related to the PROPATH statement conversion limitation which is described in file-system.od
t document.
```

Make sure you are testing with the latest P2J code. On November 16th, Costin checked in changes to EnvironmentOps, FileSystemDaemon and other files which provides full support for PROPATH assignment. The file-system.odt document has not yet been updated.

```
to return the default value of ".:". In Windows this behavior become incorrect. Looks like we need to insert t
he OS detection code into P2J
```

Yes, we should implement a Windows-specific default (what is it?) in EnvironmentOps. I don't want to have to put something in the directory.xml.

#### #25 - 12/05/2012 03:03 PM - Eugenie Lyzenko

> Yes, we should implement a Windows-specific default (what is it?) in EnvironmentOps. I don't want to have to put something in the directory.xml.

OK.

I have tested once again the file_system.odt 4GL calls, features side by side for Linux and Windows machines. And found one expected difference for OS-DRIVES call(file_system_training18.p). In Windows we return the drive list separated by comma, in Linux we return "". And this is expected according to the Progress documentation and not an issue.

The other missing point in Windows is supporting of the following attributes:

```
FILE-INFO:FILE-CREATE-DATE file_system_training30.p
FILE-INFO:FILE-CREATE-TIME file_system_training31.p
```

Currently P2J return value are unknowing for both(? value) and this is OK because in Linux the attributes are not supported. But we need to implement them in Windows because Progress Language Reference reports they should work in Windows. I'm going to check in native 4GL on Windows to find out how it works in Windows.

And I will clarify what is the default behavior for PROPATH in Windows.

#### #26 - 12/05/2012 05:35 PM - Eugenie Lyzenko

Several notes for PROPATH get/set functionality
1. The P2J converts, compile and run the PROPATH statement.
2. The implementation of the PROPATH setter is not fully correct. The internal PROPATH value must not contain platform specific path separator. The both Linux(:) and Windows(;) should be replaced with comma(,). This is what I see in native 4GL in Windows.
3. The handling of the Windows path is different in our P2J running in Windows and native 4GL in Windows. Our P2J code eats one "\" symbol from the new PROPATH to be set. For example if we trying to set

```
propath = "C:\Windows\path;"
```

the result will be

```
propath == "C:Windowspath;"
```

so it is required to double the slashes:

```
propath = "C:\\Windows\\path;"
```

to have

```
propath == "C:\Windows\path;"
```

4. The default value is neither ".:" nor ".;". It can be only "." for every OS if not defined anywhere.
5. It is not clear how the PROPATH modification is happening in 4GL. Because when application starts the propath is:
".,path1,path2,..." and after modification propath = "C:\Windows\path;" it become:
"C:\Windows\path,,path1,path2,..." so the leading "." entry is replaced with ",". So this point should be clarified additionally.

#### #27 - 12/05/2012 05:38 PM - Eugenie Lyzenko

The following features were confirmed as working in native 4GL in Windows:
FILE-INFO:FILE-CREATE-DATE file_system_training30.p
FILE-INFO:FILE-CREATE-TIME file_system_training31.p

So looks like we need to implement them for Windows environment.

#### #28 - 12/06/2012 10:53 AM - Greg Shah

```
3. The handling of the Windows path is different in our P2J running in Windows and native 4GL in Windows. Our
P2J code eats one "\" symbol from the new PROPATH to be set. For example if we trying to set

propath = "C:\Windows\path;"

the result will be

propath == "C:Windowspath;"
```

Make sure your p2j.cfg.xml is properly setup for Windows conversions.  This should not be happening.

```
2. The implementation of the PROPATH setter is not fully correct. The internal PROPATH value must not contain
platform specific path separator. The both Linux(:) and Windows(;) should be replaced with comma(,). This is w
hat I see in native 4GL in Windows.
```

What does the Progress documentation report on this topic?

```
5. It is not clear how the PROPATH modification is happening in 4GL. Because when application starts the propa
th is:
".,path1,path2,..." and after modification propath = "C:\Windows\path;" it become:
"C:\Windows\path,,path1,path2,..." so the leading "." entry is replaced with ",". So this point should be cla
rified additionally.
```

In Windows there are some Progress-specific paths that are always present.  We don't have any equivalent.  Could that be what you are seeing?
Please show an example including the result of the assignment as displayed on Windows.

#### #29 - 12/06/2012 02:09 PM - Eugenie Lyzenko

Make sure your p2j.cfg.xml is properly setup for Windows conversions. This should not be happening.

The p2j.cfg.xml is the same as it is used for all Windows samples conversions. Consider the following sample Progress code:

```
...
chPropath = propath.
propath = chPropath + "C:\Windows\system;C:\Users\Eugenie\bin;".
...
```

This should be OK for Progress in Windows. And the conversion result in Windows:

```
...
            chPropath.assign(EnvironmentOps.getSearchPath());
            EnvironmentOps.setSearchPath(concat(chPropath, "C:Windowssystem;C:Users\033ugenie\bin;"));
...
```

While the correct conversion result should be:

```
...
            EnvironmentOps.setSearchPath(concat(chPropath, "C:\\Windows\\system;C:Users\\Eugenie\\bin;"));
...
```

The expected result can be obtained when we modify the Progress code this way:

```
...
propath = chPropath + "C:~\Windows~\system;C:~\Users~\Eugenie~\bin;".
...
```

This code properly converted and run in P2J on Windows:

```
...
            EnvironmentOps.setSearchPath(concat(chPropath, "C:\\Windows\\system;C:Users\\Eugenie\\bin;"));
...
```

The remaining difference is the 4GL in Windows is able to handle "C:\Windows\system;C:\Users\Eugenie\bin;" case while P2J - not. But I guess this is not an issue because the Progress Language Reference tells the "~\" really means "\".

2. The Progress language reference(page 879) reports: "Progress stores the PROPATH as a comma-separated list of directories. (Progress strips the operating-specific separation characters (a colon ( : ) on UNIX; a semicolon ( ; ) on Windows) and replaces them with commas."

#### #30 - 12/06/2012 02:23 PM - Greg Shah

1. The conversion of string literals when the target is Windows, should already double up the \ characters, but in this case we have character.progressToJavaString() coded to do the right thing for Linux, not Windows.  Please look into changes there. Perhaps it needs to take a second boolean parameter named "windows".  If true, then it processes for Windows and if false, it works like today. Also, it gets called from lots of other places in the com/goldencode/p2j/ tree and from TRPL rules, so these places need update.

2. The propath processing code will need changes to handle the comma issue.

You can work on both of these.

#### #31 - 12/06/2012 02:32 PM - Eugenie Lyzenko

*- File propath_set_4gl_win.jpg added*

*- File propath_get_4gl_win.jpg added*

3. Yes, what I see in 4GL is predefined Progress-specific path. The code to display PROPATH:

```
chPropath = propath.
display chPropath.
```

will result in propath_get_4gl_win.jpg. The code to set PROPATH

```
propath = "C:~\Windows~\system;C:~\Users~\Eugenie~\bin;".
display propath.
```

or

```
propath = "C:\Windows\system;C:\Users\Eugenie\bin;".
display propath.
```

will both result in propath_set_4gl_win.jpg. Note the final value is not completely replaced with new one. The result is the sum of old PROPATH value and new string.

#### #32 - 12/06/2012 02:54 PM - Eugenie Lyzenko

You mean the current conversion "C:\Windows\system;C:\Users\Eugenie\bin;" to "C:Windowssystem;C:Users\033ugenie\bin;" is expected because the source is going via character.progressToJavaString() call and changing there(because in Linux single "\" is considering as special escape character). Am I understanding correctly?

> You can work on both of these.

OK.

#### #33 - 12/06/2012 03:22 PM - Greg Shah

```
the source is going via character.progressToJavaString() call and changing there(because in Linux single "\" i
s considering as special escape character). Am I understanding correctly?
```

Yes, exactly.

#### #34 - 12/07/2012 05:59 PM - Eugenie Lyzenko

The fix for path separator in PROPATH handling issue.

The idea is to have the helper method fixupPropath(String propathInitial) inside the EnvironmentOps.java source or inside StringHelper.java which can be more intuitive. This static method replaces the current path separator char with comma. The path separator char is one that is used inside the P2J running on certain operating system. The path separator can be configured in directory.xml file or if not it is the current separator for the OS the P2J engine is running.

The internal representation of the PROPATH storing in EnvironmentOps class after the PROPATH is changed from user code will be system neutral. The following changes are suggested for EnvironmentOps.java source:

```
...
   public static String fixupPropath(String propathInitial)
   {
      StringBuffer sb = new StringBuffer();

      for (int i = 0; i < propathInitial.length(); i++ )
         if (propathInitial.charAt(i) == getPathSeparator().charAt(0))
            sb.append(',');
         else
            sb.append(propathInitial.charAt(i));

      return sb.toString();
   }
...// Here we fix the possible incorrect chars while setting
   public static void setSearchPath(String path , boolean conversion)
   {
      if (path != null)
      {
         work.obtain().propath = fixupPropath(path);
...
      }
   }
```

```
...// Changing the default value to the current system path separator, instead of ":"
   public static String getPathSeparator()
   {
      return Utils.getDirectoryNodeString(null, "path-separator",
                                          File.pathSeparator);
   }
...
   public static String getLegacyPathSeparator()
   {
      return Utils.getDirectoryNodeString(null,
                                          "file-system/path-separator",
                                          File.pathSeparator);
   }
...// Another changing of the default value from "/" to current system one
   public static String getLegacyFileSeparator()
   {
      return Utils.getDirectoryNodeString(null,
                                          "file-system/file-separator",
                                          File.separator);
   }
...
```

The question: I consider the path separator is one char symbol. Do we need to consider possible cases of the 2 chars more path separators or this does no make a sense?

#### #35 - 12/10/2012 08:24 AM - Greg Shah

I am OK with the changes to setSearchPath(), getPathSeparator(), getLegacyPathSeparator() and getLegacyFileSeparator().

The fixupPropath() should be a little different. I don't want getPathSeparator() called more than once, because it will have negative performance consequences and there is no reason to honor any changes to it during the method.

In addition, although it is safe enough (for now) to assume a single character path separator, I prefer to make it safe for the future. It seems to me that we can use String.replaceAll() or we can simply use String.indexOf(String, int) in our own loop. replaceAll() yields less code (which is good) but if the path separator ever contains characters with special meaning in a regex, then it would have problems. So the indexOf(String, int) approach would solve that problem.

#### #36 - 12/10/2012 09:16 AM - Eugenie Lyzenko

> The fixupPropath() should be a little different. I don't want getPathSeparator() called more than once, because it will have negative performance consequences and there is no reason to honor any changes to it during the method.

I see. What if we will have internal variable as buffer, say pathSeparator which initially be null. When getPathSeparator() called for the first time - it will do the real work, the next calls will just return the buffered value.

> In addition, although it is safe enough (for now) to assume a single character path separator, I prefer to make it safe for the future. It seems to me that we can use String.replaceAll() or we can simply use String.indexOf(String, int) in our own loop. replaceAll() yields less code (which is good) but if the path separator ever contains characters with special meaning in a regex, then it would have problems. So the indexOf(String, int) approach would solve that problem.

OK. I'll rework this code.

#### #37 - 12/10/2012 09:58 AM - Greg Shah

```
I see. What if we will have internal variable as buffer, say pathSeparator which initially be null. When getPa
thSeparator() called for the first time - it will do the real work, the next calls will just return the buffer
ed value.
```

That is not exactly what I mean.  Just use a local variable in fixupPropath() and only call getPathSeparator() once:

```
public static String fixupPropath(String propath)
{
   String pathSep = getPathSeparator();
   ...the rest of fixupPropath()...
```

#### #38 - 12/10/2012 05:39 PM - Eugenie Lyzenko

The new version of the fixupPropath() method.
1. Handles the arbitrary length of the path separator
2. Optimizing for usage of the getPathSeparator()
3. The loop logic has been changed. I do not use the regular expression calls line replaceAll() due to the possible incompatibility issues with chars in path separator. The idea is to find elements between path separator within new PROPATH string, extract them separating with comma.

Works OK in Linux, tested with possible critical cases like "", ":", "::". Currently is under testing on Windows.

```
   public static String fixupPropath( String propathInitial )
   {
      String pathSep = getPathSeparator();
      StringBuffer sb = new StringBuffer();
      int pathSepLength = pathSep.length();

      // The starting position of the next PROPATH element
      int nextSegment = 0;
      // Scan input parameter for the next occurence of the path separator.
      // The next PROPATH element starting point is the sum of the current
      // PROPATH element and length of the path sepatator
      for (int i = 0;
           (i = propathInitial.indexOf(pathSep, nextSegment)) >= 0;
           nextSegment = i + pathSepLength)
      {
         // Extract the PROPATH entry up to the nearest path separator
         sb.append(propathInitial.substring(nextSegment, i));
         // Replace path separator with comma
         sb.append(',');
      }
      // We have to handle the case when new PROPATH sting is not finished
      // with path separator.
      if (nextSegment < propathInitial.length())
         sb.append(propathInitial.substring(nextSegment));

      return sb.toString();
   }
```

#### #39 - 12/11/2012 08:22 AM - Greg Shah

Looks good.

#### #40 - 12/11/2012 09:08 AM - Eugenie Lyzenko

The Windows tests is OK too. I would like to make more tests with simulated two chars path separator to be sure this case is handling correctly as well.

Another point. As you know the "~" char is the special one. So to add to the PROPATH in Linux value like "~/bin" the developer should describe the modified propath string as "~~/home". This is described in Progress Language Reference book and exactly the same as our conversion works. Not an issue.

One more question. What is the testing/submitting plan for this change? Am I combining it with the fix for "\" handling in Windows I'm currently working on to have single update for two features? Or we need these two fixes to be applied separately?

#### #41 - 12/11/2012 09:22 AM - Greg Shah

This is all a single update.

#### #42 - 12/11/2012 03:32 PM - Eugenie Lyzenko

Fix for backslash processing for conversion on Windows.

The suggested solution will be activated only when running conversion on Windows. The Linux processing remains untouched. The only change is class character, file character.java. Method progressToJavaString(...), line ~1661:

```
   public static String progressToJavaString(String progress)
...
            case '~':
            case '\\':
               // This addition handles the '\' character different way for Windows
               // using new method from EnvironmentOps class to detect
               // the underlying OS.
               if (EnvironmentOps.inWindows() && current == '\\')
               {
                  sb.append("\\\\");
                  break;
               }

               // there must always be a following character
               i++;
               next = readChar(contents, i);

               // handle the special cases that don't have a one-to-one
               // correspondence in Java
               if (next == 'E')
               {
                  // the special case for the ESC char
                  sb.append("\\033");
               }
...
```

The change was tested under Windows and confirmed as working for both single "\" chars and double backslashes as indicator of the network locations like this: "\\WIN_SERVER\Shared\Directory\On\Server".

#### #43 - 12/11/2012 05:27 PM - Greg Shah

Do not use the current conversion platform to determine how we convert strings. We can't go this route. The problem is that the way we convert strings must be controlled by a separate configuration flag. That flag will be passed in as a boolean parameter. Please see my comment in https://projsrv01/redmine/issues/1648#note-30 about the 2nd boolean "windows" parameter.

#### #44 - 12/12/2012 01:37 PM - Eugenie Lyzenko

> Do not use the current conversion platform to determine how we convert strings. We can't go this route.

OK. I see. Another words we should have the configurable parameter like "target-windows" either inside the p2j.cfg.xml file or as command option for ConversionDriver main worker to be able to make the Windows target conversion on Linux system. Then depending on this parameter we go the special way in character.progressToJavaString(String progress, boolean windows). Correct?

I've made two versions of character.progressToJavaString():

```
...
// This is the old style call
   public static String progressToJavaString(String progress)
   throws IllegalArgumentException
   {
      return progressToJavaString(progress, false);
   }
// The updated version
   public static String progressToJavaString(String progress, boolean windows)
   throws IllegalArgumentException
   {
...
            case '~':
            case '\\':
               if (windows && current == '\\')
               {
                  sb.append("\\\\");
                  break;
               }
               // there must always be a following character
               i++;
               next = readChar(contents, i);
...
   }
...
```

The other places where the method character.progressToJavaString() is called are:

```
com/goldencode/p2j/uast/Variable.java
com/goldencode/p2j/uast/ProgressParser.java
com/goldencode/p2j/uast/ExpressionEvaluator.java
com/goldencode/p2j/convert/ExpressionConversionWorker.java
com/goldencode/p2j/convert/UnreachableCodeWorker.java
```

Now I'm finding one that controls the string literal like one from considering example. The suggested candidate: com/goldencode/p2j/convert/ExpressionConversionWorker.java

```
...
      public String progressToJavaString(String text)
      {
         return character.progressToJavaString(text,
                         Configuration.getParameter("target-windows", false));
      }
...
```

With this change we could have the proper handling of the Windows "\" chars converting on Linux for Windows target system.

#### #45 - 12/12/2012 02:33 PM - Greg Shah

Your latest approach is all good except that we don't need a new p2j.cfg.xml parameter.  We already have one that is exactly what we need: "unix-escapes".  If set to true, then \ should be treated as an escape character.  If false, then the \ to \\ conversion should be done, like in Windows.

Just change this:

```
Configuration.getParameter("target-windows", false)
```

to this:

```
Configuration.getParameter("unix-escapes", true)
```

#### #46 - 12/13/2012 06:33 PM - Eugenie Lyzenko

Testing on both Linux and Windows reports the changes in
com/goldencode/p2j/util/character.java
com/goldencode/p2j/convert/ExpressionConversionWorker.java

as I noted above is enough and only enough to get properly converted Windows style path string taking into account the "unix-escapes" configuration parameter. I'm preparing the update for upload. But I think the full regression testing is mandatory to safely apply this change.

#### #47 - 01/07/2013 01:26 PM - Greg Shah

Please upload the complete update zip into this task so that I can do a final code review.

#### #48 - 01/07/2013 01:51 PM - Eugenie Lyzenko

*- File evl_upd20121214a.zip added*

The cumulative fix has been uploaded here.

#### #49 - 01/07/2013 02:54 PM - Greg Shah

Feedback:

1. In ExpressionConversionWorker, instead of this:

```
import com.goldencode.p2j.cfg.Configuration;
```

please use this:

```
import com.goldencode.p2j.cfg.*;
```

2. In character.java, please merge with the latest version in Bazaar.  The 069 and 070 changes are missing from your version.

3. In EnvironmentOps, please update getLegacyCaseSensitive() to set the default based on the current OS.

Then please upload the final version of the changes to this task.

#### #50 - 01/07/2013 03:47 PM - Eugenie Lyzenko

*- File evl_upd20130107a.zip added*

The new archive has been uploaded here for you to review.

The question: In getLegacyCaseSensitive() I have used "inline" code to check if the OS is Windows family. May be we need separate method to do this, something like isUnderWindowsFamily() to simplify modifications? I used the similar code already 3 times in P2J source tree and separate method in EnvironmentOps() could be better? The other 2 times - in StringHelper class.

#### #51 - 01/07/2013 04:33 PM - Greg Shah

Yes, please make a new method for isUnderWindowsFamily() and use that everywhere necessary.  Then upload the latest code.

#### #52 - 01/07/2013 05:22 PM - Eugenie Lyzenko

*- File evl_upd20130107b.zip added*

The Windows detection code has been moved to separate statis method of the EnvironmentOps class.

Also I made one small syntax fix for Javadoc in character.java. The parameter for quoterWorker() is really customQuote. I just found new warning in Javadoc generation process.

#### #53 - 01/08/2013 09:23 AM - Greg Shah

Everything looks good except for 1 thing.

We don't want com.goldencode.util to depend upon com.goldencode.p2j.util.  The idea is that the com.goldencode.util is independent and could even be in another jar file (used by multiple projects).  It is expected that the com.goldencode.p2j.util will depend upon com.goldencode.util, but no dependencies in the other direction.

Please create a new class com.goldencode.util.PlatformHelper.  Put the static method isWindowsFamily() in there.  Call it from StringHelper and from com.goldencode.p2j.util.EnvironmentOps.

#### #54 - 01/08/2013 10:13 AM - Eugenie Lyzenko

Please create a new class com.goldencode.util.PlatformHelper. Put the static method isWindowsFamily() in there. Call it from StringHelper and from com.goldencode.p2j.util.EnvironmentOps.

OK. What about caching the Windows family flag inside PlatformHelper class? Something like:

private static boolean(or logical) inWindows = not defined; // Not yet defined

If the first call to isUnderWindowsFamily() then we set up the cahced value first:
inWindows = Full detection code

On the next calls we can just return already defined cached value instead of making System.getProperty("os.name").toLowerCase().indexOf("win") every time.

#### #55 - 01/08/2013 10:17 AM - Greg Shah

Yes, that makes sense.

#### #56 - 01/08/2013 10:51 AM - Eugenie Lyzenko

*- File evl_upd20130108a.zip added*

The new update to review. The method isUnderWindowsFamily() has been move to the separate class com.goldencode.util.PlatformHelper.

#### #57 - 01/08/2013 11:49 AM - Greg Shah

Feedback on the PlatformHelper:

1. Remove the JPRM column from the header. Since there are no JPRM numbers and JPRM no longer exists, there is no need for that column.

2. Remove the @version number.  The history is enough detail on the version.

3. I think we can use a simpler approach:

```
   /** Variable to cache the current OS Windows family value. */
   private static boolean winFamily =
      System.getProperty("os.name").toLowerCase().indexOf("win") >= 0;

   /**
    * Checks is the current OS the P2J is running on belongs to the Windows
    * family.
    *
    * @return   true if the current OS is from Windows family or
    *           false otherwise.
    */
   public static boolean isUnderWindowsFamily()
   {
      return winFamily;
   }
```

#### #58 - 01/08/2013 12:18 PM - Eugenie Lyzenko

*- File evl_upd20130108b.zip added*

OK. The update has been uploaded.

#### #59 - 01/08/2013 12:23 PM - Greg Shah

It looks good.  Please go ahead with testing.  It needs both conversion AND runtime testing.  The conversion testing is needed because the StringHelper, EnvironmentOps and ExpressionConversionWorker changes all affect the conversion processing.  Let me know if there are any questions.

#### #60 - 01/08/2013 02:47 PM - Eugenie Lyzenko

> Let me know if there are any questions.

The plan is:
After current staging passes the regression testing I will copy the current staging build to some place on lightning, apply this change, reconvert and make new harness cycle. Is it OK? I guess this will be better than doing this in staging itself.

#### #61 - 01/08/2013 03:21 PM - Greg Shah

Yes, good plan.  Make sure to compare the converted output from staging with the output from your copied build.  It should be identical.

#### #62 - 01/08/2013 04:01 PM - Eugenie Lyzenko

Is it enough to compare just src/aero/timco/majic to make sure the generated java code is identical? Or we need to compare the src/syman directories?

#### #63 - 01/08/2013 04:18 PM - Greg Shah

You only need to compare the src/aero/timco/majic directories.

#### #64 - 01/09/2013 03:10 PM - Eugenie Lyzenko

> You only need to compare the src/aero/timco/majic directories.

The directories are identical. The next step I'm going to do is to run harness against new build.

#### #65 - 01/14/2013 01:10 PM - Greg Shah

What is the status of your update?  Did it pass harness testing?

You will need to merge it with the latest level of code in bzr (at least the ExpressionConversionWorker will have conflicts).  Then I want to get it applied to staging and tested there, along with a couple of other updates.  I'd like to get this into testing today if possible.

#### #66 - 01/14/2013 01:17 PM - Eugenie Lyzenko

> What is the status of your update? Did it pass harness testing?

The CTRL-C tests have been passed. The main part is currently in progress.

> You will need to merge it with the latest level of code in bzr (at least the ExpressionConversionWorker will have conflicts).

OK.

> Then I want to get it applied to staging and tested there, along with a couple of other updates. I'd like to get this into testing today if possible.

If the harness will be OK I think the merged update will be ready today. I would like to test my changes separately to simplify the possible regression location.

#### #67 - 01/14/2013 02:38 PM - Eugenie Lyzenko

*- File evl_upd20130114a.zip added*

The uploaded file contains the merge with the current p2j bzr status.

#### #68 - 01/14/2013 02:51 PM - Greg Shah

Are you sure you want to test this separate from other changes?  If it has already passed testing, then it should be safe.

We are already ready to run the harness in staging.  It would be a good time to add your changes.

#### #69 - 01/14/2013 03:02 PM - Eugenie Lyzenko

> Are you sure you want to test this separate from other changes? If it has already passed testing, then it should be safe.

Yes, the conversion already passed the tests but P2J runtime not completely. Can you wait for 30 min more for me to have the recent harness result from my own directory?

> We are already ready to run the harness in staging. It would be a good time to add your changes.

I have uploaded my change to staging/p2j. I can unzip it now. Then you will need to rebuild P2J and reconvert(I do not have enough permissions to do this). Then I'll start DB restore and harness. Is this plan OK?

#### #70 - 01/14/2013 03:07 PM - Eugenie Lyzenko

Another question. What about customer_libs subdirectory? Do we need to clean it up before testing?

#### #71 - 01/14/2013 03:16 PM - Greg Shah

Yes, we will wait 30 minutes for your previous harness results.  Let me know.  I can apply your change and rebuild if it is ready.

Yes, we need to remove the extra customer_libs to avoid the logging problem.

#### #72 - 01/14/2013 03:24 PM - Eugenie Lyzenko

> I can apply your change and rebuild if it is ready.

I have applied my update.
...
~~rw r r~~  1 evl gc 113230 Jan 14 14:41 /gc/convert/staging/p2j/evl_upd20130114a.zip
...
Please rebuild the P2J.

> Yes, we need to remove the extra customer_libs to avoid the logging problem.

OK.

#### #73 - 01/14/2013 03:41 PM - Eugenie Lyzenko

Greg,

Please start the P2J rebuild and reconversion now. After this is finished I'll start the DB restore and new harness in staging.

#### #74 - 01/14/2013 03:44 PM - Greg Shah

I just finished rebuilding with all updates applied.  Eric will be handling the reconversion in 20 minutes.  He has to back up the most recent staging conversion results first.  He will notify you when it is ready to run the harness.

#### #75 - 01/14/2013 04:29 PM - Eugenie Lyzenko

Status update: My recent update completely passed the harness cycle with the last testing in separate directory.

#### #76 - 01/16/2013 09:24 AM - Eugenie Lyzenko

*- File evl_upd20130116a.zip added*

Updated archive uploaded.

#### #77 - 04/09/2013 04:46 PM - Greg Shah

The final work with this task was checked in as bzr revision 10152 on January 17, 2013.  As far as I know this task may be complete.  I did look through it and found this one thing that I am not sure of:

> 5. It is not clear how the PROPATH modification is happening in 4GL. Because when application starts the propath is:

"."path1,path2,..." and after modification propath = "C:\Windows\path;" it become:
"C:\Windows\path,,path1,path2,..." so the leading "." entry is replaced with ",". So this point should be clarified additionally.

Is this done?

Also: are there any other things that need to be done before this task is considered closed?

#### #78 - 04/09/2013 07:09 PM - Eugenie Lyzenko

> Is this done?

I have clarified this point. Both Windows and Linux 4GL systems shows the same behavior:

```
propath = "NewPropath".
display propath.
```

shows the sum of the new propath value with the old one. And new entries become in the head of the resulting string:
NewPropath,OldPropath

The update as of January 17, 2013 does not work this way. Currently we just substitute the old value with the new one.

Also: are there any other things that need to be done before this task is considered closed?

There are another statements mentioned here are not currently supported in Windows:

```
INPUT(INPUT-OUTPUT) THROUGH
```

However they are related to the process handling and native library implementation so it is not directly related to this task I guess.

#### #79 - 04/10/2013 08:09 AM - Greg Shah

OK, after you are finished with #1650, you can fix the propath assignment behavior.  When that is done we will close this task.

#### #80 - 04/25/2013 09:50 AM - Greg Shah

*- Status changed from WIP to Closed*

*- % Done changed from 0 to 100*

#### #81 - 11/16/2016 11:43 AM - Greg Shah

*- Target version changed from Milestone 7 to Runtime Support for Server Features*

### Files

| | | | |
|---|---|---|---|
| propath_get_4gl_win.jpg | 47.3 KB | 12/06/2012 | Eugenie Lyzenko |
| propath_set_4gl_win.jpg | 41.7 KB | 12/06/2012 | Eugenie Lyzenko |
| evl_upd20121214a.zip | 97.6 KB | 01/07/2013 | Eugenie Lyzenko |
| evl_upd20130107a.zip | 99.3 KB | 01/07/2013 | Eugenie Lyzenko |
| evl_upd20130107b.zip | 109 KB | 01/07/2013 | Eugenie Lyzenko |
| evl_upd20130108a.zip | 110 KB | 01/08/2013 | Eugenie Lyzenko |
| evl_upd20130108b.zip | 110 KB | 01/08/2013 | Eugenie Lyzenko |
| evl_upd20130114a.zip | 111 KB | 01/14/2013 | Eugenie Lyzenko |
| evl_upd20130116a.zip | 111 KB | 01/16/2013 | Eugenie Lyzenko |