Base Language - Feature #1649

implement Windows environment/registry access

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

Status:	Closed	**Start date:**	01/15/2013
Priority:	Normal	**Due date:**	05/17/2013
Assignee:	Eugenie Lyzenko	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	Runtime Support for Server Features		
billable:	No	**version:**	
vendor_id:	GCD		

Description

Subtasks:

Feature # 1960: conversion support for Windows environment/registry access	**Closed**
Feature # 1961: implement runtime backing for Windows environment/registry access	**Closed**

Related issues:

Related to Base Language - Feature #2385: implement environment access on GUI...	**Closed**	**09/04/2014**

History

#1 - 10/21/2012 01:10 PM - Greg Shah

Priority features: GET-KEY-VALUE, PUT-KEY-VALUE, USE, LOAD, UNLOAD

The COLOR and FONT options to PUT-KEY-VALUE do not need to be supported at this time.

#2 - 10/31/2012 01:32 PM - Greg Shah

- Target version set to Milestone 12

#3 - 10/31/2012 01:33 PM - Greg Shah

- Target version changed from Milestone 12 to Milestone 7

#4 - 01/15/2013 04:22 PM - Greg Shah

- Estimated time changed from 40.00 to 80.00

#5 - 01/15/2013 04:28 PM - Greg Shah

For now, we need to add conversion support (for all features mentioned above) and the minimum runtime stubs to make the converted code compile. The stub methods can be placed in EnvironmentOps. The work on the real runtime backing code will be done later.

#6 - 01/17/2013 06:27 PM - Eugenie Lyzenko

Greg,

I have followed with the conversion process and found the language keywords are already supported in progress.g and AST, JAST generations. So the *.ast, *.jast files are seems to be correct now. And to implement conversion support we need to implement jast -> java source file generation process.

This is a new implementation task for me and have the questions:
1. I need to write the conversion rules to implement this task?
2. Add stub methods to the EnvironmentOps class to be able to compile conversions?
3. The rule I need to change is: convert/language_statements.rules?

Please let me know if I'm wrong or missed for something.

#7 - 01/17/2013 06:44 PM - Greg Shah

```
And to implement conversion support we need to implement jast -> java source file generation process.
```

No, as far as I know there is no need for work to translate the JAST into Java. The features needed by the conversion should already be there and convert/brew.xml does the source file generation.

```
1. I need to write the conversion rules to implement this task?
```

Yes. Your rules need to create the nodes in the JAST tree, which will then emit as source code that calls the backing runtime methods.

```
2. Add stub methods to the EnvironmentOps class to be able to compile conversions?
```

Yes.

```
3. The rule I need to change is: convert/language_statements.rules?
```

Yes.

#8 - 01/23/2013 09:29 AM - Eugenie Lyzenko

Currently there are no significant issues with the statements:
PUT-KEY-VALUE, USE, LOAD, UNLOAD

The small question for these statements is: We need the P2J backend to support both input types, character and String, right. I'm planning to implement the real java code inside for example setKeyValue(String, String, String). The other cases for this call will just use String based "master" like this: setKeyValue(String, character, String)-> transforms character to String by getValue() and calls setKeyValue(String, String, String). Right?

The main issue I'm currently working on is the GET-KEY-VALUE. The Progress uses C-style parameter. This is not a function. like value = GET-KEY-VALUE, this is the statement: GET-KEY-VALUE. The Value is the variable to store the returned value. So we need to transform from the 4GL style: GET-KEY-VALUE to Java style: Value = GET-KEY-VALUE. This is a kind of assignment. I'm going to insert the appropriate modifications into convert/language_statements.rules file leaving annotation assignments rules untouched. Let me know if there are objections.

#9 - 01/23/2013 10:04 AM - Greg Shah

```
 We need the P2J backend to support both input types, character and String, right.
```

Yes.

```
I'm planning to implement the real java code inside for example setKeyValue(String, String, String). The other
 cases for this call will just use String based "master" like this: setKeyValue(String, character, String)-> t
ransforms character to String by getValue() and calls setKeyValue(String, String, String). Right?
```

Good. The only thing here is to put a // TODO implement this comment inside the real worker (setKeyValue(String, String, String)), since we are just stubbing it for now. You can go ahead and write the others to call setKeyValue(String, String, String) since that doesn't take long.

```
So we need to transform from the 4GL style: GET-KEY-VALUE to Java style: Value = GET-KEY-VALUE. This is a kind
 of assignment. I'm going to insert the appropriate modifications into convert/language_statements.rules file
leaving annotation assignments rules untouched. Let me know if there are objections.
```

Yes. Generally, in such cases we use a template to help us get the tree structure right. I recommend using "simple_unfinished_assign". It can be done something like this:

```
            <action>
               lastid = tw.graft("simple_unfinished_assign",
                                 copy,
                                 closestPeerId,
                                 "javaname",
                                 javaname).id
            </action>
            <action>
               createPeerAst(java.static_method_call, "EnvironmentOps.getKeyValue", lastid)
            </action>
```

You would have to set the javaname variable to the converted name of the variable to assign AND you would have to ensure that it did NOT emit again during normal processing (which it would in variable_references.rules). Most often we hide the variable during annotations processing AFTER we have copied the javaname information into the main statement's (the KW_GET_K_V node).

From there, you can let everything else emit as by default.

#10 - 01/28/2013 09:25 PM - Eugenie Lyzenko

- File registry.p.zip added

- File evl_upd20130128a.zip added

The first version to review. The modification log:
1. The file util/EnvironmentOps.java was changed adding the stubs for appropriate Progress statements.
2. The conversion support modification are located in rules/conversion/language_statements.rules
- All statements except LOAD and GET-KEY-VALUE converting by adding references to static methods like this:
<action>
createPeerAst(java.static_method_call,
'EnvironmentOps.methodToUse',
closestPeerId)
</action>
- The statement GET-KEY-VALUE requires special processing due to the fact the lat statement parameter is the value to store result and should be used as lvalue in Java code:
<variable name="javaname"   type="java.lang.String" />
<variable name="varref"   type="com.goldencode.ast.Aast" />

```
&lt;action&gt;
               varref = copy.getChildAt(2).getChildAt(0)
            &lt;/action&gt;
            &lt;rule&gt;varref.type == prog.expression
               &lt;action&gt;varref = varref.getChildAt(0)&lt;/action&gt;
            &lt;/rule&gt;
            &lt;action&gt;
               javaname = #(java.lang.String) execLib("get_javaname", varref)
            &lt;/action&gt;



&lt;action&gt;
               lastid = tw.graft("simple_unfinished_assign",
                                 copy,
                                 closestPeerId,
                                 "javaname",
                                 javaname).id
            &lt;/action&gt;
            &lt;action&gt;
               createPeerAst(java.static_method_call,
                             'EnvironmentOps.getKeyValue',
                             lastid)
            &lt;/action&gt;
            &lt;action&gt;copy.getChildAt(2).remove()&lt;/action&gt;
After the variable is moved to lvalue is should be deleted from the AST to eliminate duplication. We can make 
it hidden but in this case we still have the wrapper from string value remaining. Moreover we need this variab
le to exist at the moment of the conversion the GET-KEY-VALUE. So the note is removed completely after usage.
- The LOAD statement requires the special processing for optional NEW parameter flag. We need to emit new Java
 boolean variable:
      &lt;rule&gt;type  prog.kw_new and
            parent.type  prog.kw_load
         &lt;action&gt;createJavaAst(java.bool_true, "true", closestPeerId)&lt;/action&gt;
      &lt;/rule&gt;
```

The testcase file to convert - registry.p is also appended here.

#11 - 01/29/2013 01:04 PM - Greg Shah

Feedback on the testcase:

Please add code that shows the use of:

1. LOAD with BASEKEY "INI"

2. PUT-KEY-VALUE with "?" in section and/or key and/or value.

3. GET-KEY-VALUE with "?" in section and/or key.

I will look at the coding changes next.

#12 - 01/29/2013 01:13 PM - Greg Shah

Code feedback:

1. The stub methods in EnvironmentOps cannot call character.getValue() (et al) without some unknown protection. Solve this by just wrapping any String/boolean parameters in a wrapper constructor (new character(my_string) or new logical(my_boolean)). The real worker method should be the form that has all wrapper constructors (e.g. getKeyValue(character, character)). That is the place where we will centralize all the unknown value processing before doing any real work.

2. Please remove any extra space at the end of method signatures:

```
public void myMethod(Type parm1, Type parm2 )
                                           ^
```

Otherwise, the update looks good.

#13 - 01/29/2013 01:34 PM - Eugenie Lyzenko

> The real worker method should be the form that has all wrapper constructors (e.g. getKeyValue(character, character)).

You mean the final real worker where we will implement the code logic should be changed from e.g. getKeyValue(String, String) to getKeyValue(character, character)? All possible cases of the getKeyValue() should finally call the getKeyValue(character, character) as worker. Correct?

#14 - 01/29/2013 01:52 PM - Eugenie Lyzenko

- File registry_20130129a.p.zip added

The new testcase version has been uploaded here.

#15 - 01/29/2013 01:56 PM - Greg Shah

```
You mean the final real worker where we will implement the code logic should be changed from e.g. getKeyValue(
String, String) to getKeyValue(character, character)? All possible cases of the getKeyValue() should finally c
all the getKeyValue(character, character) as worker. Correct?
```

Yes.

#16 - 01/29/2013 02:05 PM - Greg Shah

The latest testcase is fine. Please check it into bzr (p2j_repo/testcases/uast/).

#17 - 01/29/2013 02:28 PM - Eugenie Lyzenko

Another question. In the new approach I need to reformat incomplete calls like this:
getKeyValue(String section)
We have the opportunity to transfer to call either
return getKeyValue(new character(section), new character("?"));
or
return getKeyValue(new character(section), new character());
or
return getKeyValue(new character(section), null);

We need to mark the default behavior for missing parameters. The right way is to set unknown value - new character()? Correct?

#18 - 01/29/2013 02:54 PM - Greg Shah

```
We need to mark the default behavior for missing parameters.
```

Yes.

```
The right way is to set unknown value - new character()?
```

I don't know. We can only find this by testing in the 4GL environment. For now, just put TODOs there an do the minimum to satisfy javac.

#19 - 01/29/2013 03:38 PM - Eugenie Lyzenko

- File evl_upd20130129a.zip added

Uploaded updated conversion support.

And I have implementing just (character)null approach for missing parameters. This way we can simply separate the default case from "unknown" value which can be not the same.

#20 - 01/29/2013 04:19 PM - Eugenie Lyzenko

> Please check it into bzr (p2j_repo/testcases/uast/).

Done.

#21 - 01/30/2013 12:29 PM - Greg Shah

I am OK with the changes. But there is a conflict in bzr. Please merge with the latest changes and upload it.

#22 - 01/30/2013 01:42 PM - Eugenie Lyzenko

- File evl_upd20130130a.zip added

Merged update has been uploaded.

#23 - 01/30/2013 02:44 PM - Greg Shah

OK, this looks good. Let's wait until tomorrow for conversion testing, since Constantin is in testing with another change that will conflict with language_statements.rules. You'll merge tomorrow and then do the conversion testing. Runtime regression testing won't be needed.

#24 - 01/31/2013 01:17 PM - Eugenie Lyzenko

- File evl_upd20130131a.zip added

The new update merged with recent code from Constantin and moved used variable to the head of the file because further conversion changes will require this change.

#25 - 02/03/2013 06:16 PM - Eugenie Lyzenko

- File evl_upd20130203a.zip added

Bit change to fix log entry number in header. There are no code changes, just wrong number from 008 to 018 fix.

#26 - 05/21/2013 04:41 PM - Eugenie Lyzenko

After reading some background documentation I have got the implementation plan:

1. The "environment" is a set of the variables with defined values organized in (name<->value) mapping pair. The lifecycle of the environment is: load, use, get|put-key-value and unload. Every given time only one current environment can be active. When the Progress starts on Windows it loads the default working environment from registry.

2. I suggest to use separate Java class/source file in com.goldencode.p2j.util.* package, say Environments.java or Registry.java to separate all related handling there including possible support for the multiple environments on demand. We can call the methods from EnvironmentOps to this new class.

3. The code will work under Windows only(correct?). For this the check with existing PlatformHelper.isUnderWindowsFamily() will be performed and if not in Windows - just ignore or exception raising.

4. Looks like we will need to call native code to access the real Windows registry. It is planned to be implemented in Environments.java or Registry.java by JNI to the p2j.dll. Native code to touch the registry is planned to be implemented in registry.c file of the native library source tree and be compiled with MinGW for Windows.

5. The first implementation will work with only default environment reflecting to the registry. Then we add multiple environment handling and INI files.

6. The Java internals will be implemented using HashMap to store key-value pair. When application is finished the environment should be saved back to the storage(if changed).

Is the plan OK? Continue working.

#27 - 05/22/2013 11:12 AM - Greg Shah

> 2. I suggest to use separate Java class/source file in com.goldencode.p2j.util.* package, say Environments.java or Registry.java to separate all related handling there including possible support for the multiple environments on demand. We can call the methods from EnvironmentOps to this new class.

It is important to note that the same 4GL features can be backed by either the Windows registry OR by "stanza" INI files. Both need to be supported in P2J.

The INI support can be done in pure Java, but the Windows registry will probably need native code.

Either way, the interfaces on the server that are called from the converted code should be in EnvironmentOps. The low level functions needed should be on the client side, since that is where the INI files or the registry exists. We will need to provide remote access to those functions (similar to how we do it for the FileSystemOps <--> FileSystemDaemon).

We will have to determine exactly the features needed for the registry (which could be in a helper class named Registry.java) and what features are needed for INI support (which could be in a file called StanzaIni.java). If possible (and reasonable), please make the external API for both support classes into an interface called EnvironmentAccessor. Both Registry and StanzaIni should implement EnvironmentAccessor. This way, the EnvironmentOps.load() can create an EnvironmentAccessor instance and save it off, keeping the EnvironmentOps code separate from the code that actually accesses the specific "environment".

Even though the INI file usage can be implemented in pure Java, the file itself will have to be remotely read from/written to the client.

> 3. The code will work under Windows only(correct?).

I don't know. I suspect that the stanza INI is also supported on Linux/UNIX. Please check.

> if not in Windows - just ignore or exception raising.

Please check what the 4GL does and duplicate that behavior.

4. Looks like we will need to call native code to access the real Windows registry.

Yes.

by JNI to the p2j.dll. Native code to touch the registry is planned to be implemented in registry.c file of the native library source tree and be compiled with MinGW for Windows.

Yes.

6. The Java internals will be implemented using HashMap to store key-value pair.

I'm not sure here. For the INI file usage, this may be appropriate (since we don't want to reparse the file for every access). But for the registry, I suspect we need to dynamically call the WIN32 registry APIs.

More importantly, does the 4GL immediately persist any changes to the backing registry/INI file? Or are those changes kept in memory and then saved (or not) depending on future processing?

These questions need answers before we can know the exact implementation.

#28 - 05/22/2013 11:19 AM - Greg Shah

Although this is a separate 4GL feature, the OS-GETENV is a related functionality. We currently implement this in FileSystemOps.getProperty(). Today, we call into the client (FileSystemDaemon) to query a Java "system property". In the 4GL, this is actually resolved by reading the real environment variables that are part of the client process.

As part of this task, please add native code to back FileSystemOps.getProperty() by first checking the client's environment variables and only if the requested variable is not there, to fall back and use Java system properties. This code can easily be added to p2j.dll and by using common C runtime features, I think it should be very portable. The native method can be added to FileSystemDaemon.

#29 - 05/22/2013 05:43 PM - Eugenie Lyzenko

> ...
> As part of this task, please add native code to back FileSystemOps.getProperty() by first checking the client's environment variables and only if the requested variable is not there, to fall back and use Java system properties.
> ...

OK.

#30 - 05/24/2013 01:57 PM - Eugenie Lyzenko

The suggested change for OS-GETENV support:
1. FileSystemDaemon.java:
Declare native method and load p2j:

```
...
   static
   {
      System.loadLibrary("p2j");
   }

   private native static String getEnvironmentValue(String variable);
...
```

method getProperty()
replace

```
return System.getProperty(key);
```

with

```
String result = getEnvironmentValue(key);

if (result == null || result.length() == 0)
{
   result = System.getProperty(key);
}

return result;
```

2. filesys.c common native method for all platforms:

```
JNIEXPORT jstring JNICALL 
       Java_com_goldencode_p2j_util_FileSystemDaemon_getEnvironmentValue(JNIEnv  *env,
                                                                         jclass  jc,
                                                                         jstring variable)
{
   const char *env_var = (*env)->GetStringUTFChars(env, variable, NULL);
   char * result = NULL;

   if (env_var != NULL && strlen(env_var) > 0)
   {
      result = getenv(env_var);
   }

   if (result != NULL)
   {
      return (*env)->NewStringUTF(env, result);
   }
```

```
    else
    {
        return NULL;
    }
}
```

The implementation compiled in Linux and Windows (with appropriate change in build.xml) and works fine.

#31 - 05/24/2013 02:35 PM - Greg Shah

Yes, this looks good.

#32 - 05/27/2013 03:38 PM - Eugenie Lyzenko

> More importantly, does the 4GL immediately persist any changes to the backing registry/INI file? Or are those changes kept in memory and then saved (or not) depending on future processing?

After some investigations looks like we have clear picture for how it works. According to the Progress ABL documentation (ABL Development reference - dvref.pdf). The "environment" is a unique set of system variable values. Every time in application life cycle it uses "current" set of environment. When application starts it loads the system default environment set which becomes current by default. In Windows it is stored in registry. Then the application can change the current set to the new one by load->use pair taking the new environment either from registry or from INI file. The new current environment becomes active for all frames created after new environment become current by use statement.

The memory occupied by environment remains alive until the unload statement calling. This is the opportunity to free memory or cache the environment for performance if not to call unload.

The testing performed in 4GL on Windows shows the changes are committed immediately. I mean the call to put-key-value() changes the backend either registry or ini file just after returning. There are no buffering or delays before environment changes and the committing this change to registry or ini file. For example simple test:

```
...
define var chVar as character initial "CharVar0".
put-key-value section "EvlSection" key "EvlCharKey" value chVar.
<-Here the registry has been changed
message "Press a key to finish.".
pause.
...
```

or:

```
...
define var chVar as character initial "CharVar0".
load "evl_env1".
use "evl_env1".
put-key-value section "EvlSection" key "EvlCharKey" value chVar.
<- Here the ini file evl_env1.ini has been changed
message "Press a key to finish.".
pause.
...
```

So not depending on chosen implementation approach we have to commit all environment changes immediately in appropriate backend storage(registry or ini file) to preserve the Progress 4GL application logic. This in not very optimized approach especially for INI file due to open->write->close cycle we have to do for every single put-key-value statement. But if we want 1:1 implementation - we have no choice.

On the other hand we need to support co-existence several inactive environments in memory(ones that loaded but not currently using). So we need to have some in-memory structure to store the section-key-value tree. And I think it should not be dependent on the type of the backend storage(registry or ini file).

Another point is using the default environment modification statement just silently ignoring and does not work. The load statement gives the error message:

```
The LOAD of evl_env1 in path /home/eugeniel failed. (4449)
```

Trying to use the environment:

```
The USE of evl_env1 failed. (4452)
```

#33 - 05/28/2013 07:24 PM - Greg Shah

- Status changed from New to WIP

> This in not very optimized approach especially for INI file due to open->write->close cycle we have to do for every single put-key-value statement. But if we want 1:1 implementation - we have no choice.

Yes, we must implement exactly the same behavior.

> Another point is using the default environment modification statement just silently ignoring and does not work. The load statement gives the error message:
> ...
> Trying to use the environment:

I don't understand what you mean here. Please show an example.

And a question: can the INI files be used on Linux/UNIX?

#34 - 05/28/2013 07:42 PM - Eugenie Lyzenko

> I don't understand what you mean here. Please show an example.
>
> And a question: can the INI files be used on Linux/UNIX?

I mean trying to use the code that loads INI file in Linux:

```
...
define var chVar as character initial "CharVar0".
load "evl_env1".
use "evl_env1".
...
```

This gives error in 4GL under Linux. I think this means the ini files handling are not supported in Linux because if we can not load INI file - we can not use it. According to Progress docs, these statements work only in Windows so this is expected, I just have not found the explicit note for missing Linux support.

#35 - 05/28/2013 07:47 PM - Greg Shah

OK. Then you will need to generate the same errors when running on systems other than Windows.

#36 - 06/05/2013 04:46 PM - Eugenie Lyzenko

- File evl_upd20130605a.zip added

The first high level design upload for you to review. The offered implementation:
1. The class EnvironmentOps calls methods from new class Environments introduced to separate INI/Registry related methods.
2. The class Environments requests the client for implementation of the EnvironmentAccessor interface.
3. The implementation of the EnvironmentAccessor is EnvironmentDaemon - the client running daemon to be started on the ThinClient startup with other client daemons.
4. The EnvironmentDaemon is the key point of the client side to handle the INI/Registry related requests. It contains the member representing the current instance of the particular worker class implementation(either INI or registry related). Initially this member maps to the Registry java class meaning we take the environment variables from Windows registry if we do not use LOAD and USE statements.
5. The low level classes Registry and StanzaIni have been introduced to handle particular current environments. Both of them can be the current member of the EnvironmentDaemon depending on what task we currently need - INI or Registry.
6. The Registry class is planned to call native methods from P2J.DLL library to work with real Windows registry.
7. The StanzaIni is planned to be implemented in pure Java.
8. In both cases all environment changed are planned to be committed immediately.

This is the current approach in general. Continue working on implementation details.

#37 - 06/05/2013 05:43 PM - Greg Shah

The overall design looks good.

Some thoughts:

1. Please create some non-trivial samples that use a complete range of the features.

2. Make sure these samples include code that exploits the boundary conditions of this feature set. There should be examples of each parameter being unknown value and/or values that are invalid so that the full error processing can be understood. Please see testcases/uast/memptr/memptr_start_get_before_beginning.p for an example of how I have done it recently.

3. Based on this, you should be able to write proper implementations of each method.

4. Don't forget that the ThinClient constructor will need to be updated to create (and save off) a new instance of EnvironmentDaemon. Otherwise the code won't work!

#38 - 06/10/2013 05:27 PM - Eugenie Lyzenko

- File evl_upd20130610a.zip added

The new drop includes:
1. Some rework for Java implementation of base logic.
2. The ThinClient starts the EnvironmentDaemon now.
3. The StanzaIni class base logic to load and save INI file has been added.

Now I'm planning to prepare testcases to cover all possible input option set situations for statements, test on Windows system and add native implementations. Continue working.

#39 - 06/13/2013 06:58 PM - Greg Shah

Code Review

1. The use of EnvironmentDaemon.envMap should probably have keys lowercased. The change is needed in the load() method where it is put() and in the use() method where get() is called. I say this because the 4GL is usually case sensitive. Make sure your testcases check this.

2. The approach with the StanzaIni class has 2 potential design issues:

- The algorithm to read the file assumes that there is little to no whitespace in the structure of a given line. For example, a new section is always assumed to have the [character at the index position 0. What if the 4GL allows whitespace in front? There are other things like this in the current implementation. It will probably need to be refined to make it much more tolerant of structural differences.
- The approach to saving changes completely re-writes the file. This may cause the file's whitespace to be very different from the original (multiple blank lines would be condensed, extra whitespace on lines with content will be eliminated and so forth). Only if the 4GL causes this same behavior, can we use this approach.

Otherwise, the changes look good. Keep going.

#40 - 06/14/2013 03:04 PM - Eugenie Lyzenko

- File testcases_environments_20130614a.zip added

- File evl_upd20130614a.zip added

INI files findings:

1. LOAD is case insensitive, we can use evl_env for eVl_eNV.ini file, or eVl_eNV for evl_env.ini file. So we can and I think it is reasonable to convert environment name to be lowercased because really 4GL works in these examples with only object not depending on letters case.
2. The section names has constraints. It is not possible to have the section with [located not in 0 index position. Otherwise the 4GL consider the section is missing.
3. It is not possible to have the section name with leading/trailing spaces. Like this: [BadName] or [BadName] or [BadName].
4. But it is OK to have spaces inside the name: [Good Name].
5. The key names behaves the same as section name. The only possible space is inside the key name:
Valid KeyName=KeyValue
6. Leading spaces in key value will be omitted while writing the INI file with PUT-KEY-VALUE statement. The trailing ones - need more investigation because I have mutual exclusive results so more testcases required.
7. There are no blank lines between section when INI file is writing:
[Section one]
Body
[Section name]
Body

Continue working for registry findings/implementation.

#41 - 06/18/2013 07:27 PM - Eugenie Lyzenko

- File evl_upd20130618a.zip added

Ini file findings:

LOAD creates the new empty INI file.

Registry findings:

LOAD and USE does not really touch the registry for creation keys with given environments. Then PUT-KEY-VALUE creates keys within HKEY_CURRENT_USER base registry section.

LOAD env NEW
Creates the registry entry under HKEY_CURRENT_USER.
LOAD env
Opens existed entry under HKEY_CURRENT_USER or displays the message is the environment name key does not exist.

The uploaded drop for your review contains the first version of the registry related native code. Everything is happening under HKEY_CURRENT_USER branch. The Windows 4GL testings does not allow me to work with other registry sections on the same level as HKEY_CURRENT_USER. The calls from Windows API to get/set values in registry was chosen to support beginning from Windows 2000 OS version. More simple calls will not work in XP and this can be problematic for some customers machines.

Probably more investigation is required. Also the code need to be deeply tested for different edge conditions to be able to cover all possible input options combinations. Continue working here.

#42 - 06/19/2013 01:48 PM - Greg Shah

Code Review (0618a)

1. Please check if whitespace at the end of strings is ignored for environment names. In the 4GL the string comparison is not only case-insensitive, but whitespace is trimmed from the end of the strings before comparison. This would affect both the EnvironmentDaemon.load() and EnvironmentDaemon.use().

2. The use of String.concat() in StanzaIni.createEmptyIniFile() is broken. concat() returns the modified string it does not modify the contents of the String instance since String is immutable. So the code would have to look more like fileName = fileName.concat(something). Please note that it would be cleaner/faster to use StringBuilder.

3. Does the 4GL always generate a lowercase filename for all newly created stanza INI files? We need to match its case behavior exactly.

4. Both registry.c functions seem to have some common code. You may be able to abstract that code into some helper functions to make the code have less code duplication and be more readable.

Good stuff. Keep going.

#43 - 06/19/2013 01:49 PM - Greg Shah

> LOAD and USE does not really touch the registry for creation keys with given environments.

Can you explain what you mean by this?

#44 - 06/19/2013 01:58 PM - Eugenie Lyzenko

> > LOAD and USE does not really touch the registry for creation keys with given environments.
>
> Can you explain what you mean by this?

The actual registry modification is happening in PUT-KEY-VALUE statement.

The real creation for LOAD env NEW statement is happening only after first PUT-KEY-VALUE statement. In case of using:
...
LOAD env NEW.
USE env.
UNLOAD env.
...

nothing is really created and the registry remains untouched.

#45 - 06/19/2013 03:36 PM - Eugenie Lyzenko

> 1. Please check if whitespace at the end of strings is ignored for environment names. In the 4GL the string comparison is not only case-insensitive, but whitespace is trimmed from the end of the strings before comparison. This would affect both the EnvironmentDaemon.load() and EnvironmentDaemon.use().

OK. I'll check this point.

> 3. Does the 4GL always generate a lowercase filename for all newly created stanza INI files? We need to match its case behavior exactly.

No, the code

```
load "EvL_eNV2" new base-key "INI".
```

creates the file EvL_eNV2.ini in the working directory. No lowercase conversion for file name before file creation operation.

> 4. Both registry.c functions seem to have some common code. You may be able to abstract that code into some helper functions to make the code have less code duplication and be more readable.

OK.

#46 - 06/20/2013 02:05 PM - Eugenie Lyzenko

> 1. Please check if whitespace at the end of strings is ignored for environment names. In the 4GL the string comparison is not only case-insensitive, but whitespace is trimmed from the end of the strings before comparison. This would affect both the EnvironmentDaemon.load() and EnvironmentDaemon.use().

The strange result I've got in testing. The environment names "env" and "env " are absolutely different from the statement LOAD "env" and LOAD "env " point of view. Meaning:
1. The file env.ini can not be loaded via

```
LOAD "env  " base-key "INI".
```

2. The registry entry "env" can not be loaded via

```
LOAD "env  ".
```

3. Statement LOAD "env " new base-key "INI" creates new ini file

```
"env  .ini"(with extra spaces)
```

4. Statement LOAD "env " new creates new registry entry with name

```
"env  "(with extra spaces).
```

5. The file

```
"env  .ini"
```

can not be loaded via LOAD "env" base-key "INI".
6. The registry entry

```
"env  "
```

can not be loaded via LOAD "env".
7. The same difference for USE statement. If environment is loaded as "env" it can not be used by

```
USE "env  ".
```

8. Even the number of white spaces char does matter. These two environments are different:

```
"env "
```

and

```
"env  "
```

So looks like 4GL in Windows consider whitespaces at the end as meaningfull.

#47 - 06/20/2013 02:56 PM - Greg Shah

> So looks like 4GL in Windows consider whitespaces at the end as meaningfull.

This seems to be true for the LOAD and USE cases only. As far as I understand it, this does not extend to general character type comparisons, which are documented in the 4GL to be insensitive of whitespace at the end of the text.

#48 - 06/20/2013 03:30 PM - Eugenie Lyzenko

> > So looks like 4GL in Windows consider whitespaces at the end as meaningfull.
>
> This seems to be true for the LOAD and USE cases only. As far as I understand it, this does not extend to general character type comparisons, which are documented in the 4GL to be insensitive of whitespace at the end of the text.

Yes, I did mean usage within LOAD and USE statements only.

#49 - 06/21/2013 05:20 PM - Eugenie Lyzenko

- File evl_upd20130621a.zip added

- File testcases_environments_20130621a.zip added

Possible situations with registry load/get/put:
1. Registry already has the key, using LOAD env. - Environment can be loaded and used without errors.
2. Registry does not have the key, using LOAD env. - Error generation:
3. Registry already has the keys, using LOAD env NEW. - The previously keys are not deleted with (PUT|GET)-KEY-VALUE. This additionally means the LOAD env NEW statement does not clean old registry keys when we use NEW option to create new environment in registry.
4. Registry does not have the key, using LOAD env NEW. - The new environment is creating only with first PUT-KEY-VALUE.

Also if the key name in statements (PUT|GET)-KEY-VALUE is DEFAULT meaning in Java we have key == null or empty key the "default" key name is used in registry. The Windows has so called "default" key name for subkey with arbitrary value to set to handle this situation.

The new drop and testcases have been uploaded. New in this version:
- Fixed notes previously found and small bugs. Common code is moved to separated methods, functions in both C and Java code.
- The support for empty key value has been added meaning using DEFAULT key in Windows registry.
- The key creation native code has been added.
- The native code to handle Windows registry has been tested in real system.
- More testcases have been added.
- I have added some log code in registry.c for debugging purpose. Will be cleaned up after debug.

Continue working.

#50 - 06/24/2013 07:59 PM - Eugenie Lyzenko

More INI files interesting findings:
GET_KEY_VALUE for missing entry gives empty string, not error or NULL string.
PUT_KEY_VALUE for value with spaces gives writing of the extra spaces to disk leading and trailing as well, GET/PUT this value again trims extra spaces.
PUT-KEY-VALUE trims any leading and trailing spaces from the key names and write them without errors.
PUT-KEY-VALUE trims any leading and trailing spaces from the section names and write them without errors.

GET-KEY-VALUE trims any extra spaces from key value while reading from INI file if such white spaces in value was in INI file. So writing the same value again by PUT-KEY-VALUE gives truncated value. As an example consider the INI lines:

```
"EvlCharKey1=  Char Var 1 "
"EvlCharKey2=  Char Var 1"
"EvlCharKey3=Char Var 1 "
```

The " char is used here to show the line start-end. The following 4GL statements:

```
get-key-value section "EvlSection" key "EvlCharKey1" value chVar.
get-key-value section "EvlSection" key "EvlCharKey2" value chVar1.
get-key-value section "EvlSection" key "EvlCharKey3" value chVar2.
put-key-value section "EvlSectionNew" key "EvlCharKey1" value chVar.
put-key-value section "EvlSectionNew" key "EvlCharKey2" value chVar1.
put-key-value section "EvlSectionNew" key "EvlCharKey3" value chVar2.
```

will give the new lines:

```
"EvlCharKey1=Char Var 1"
"EvlCharKey2=Char Var 1"
"EvlCharKey3=Char Var 1"
```

GET-KEY-VALUE accept the leading/trailing spaces for key and section names, considering them without spaces"
" EvlCharKey 10 " "EvlCharKey 10 " " EvlCharKey 10" referring to the same key name
"Evl Section1 " " Evl Section1" " Evl Section1 " referring to the same section name.

The only exception/rule here is the sections and key name in files before GET-KEY-VALUE must be without extra spaces as it can be after manual modification outside the Progress system. One example is editing INI file with Notepad. Only key value can have the leading/trailing spaces in INI file.

#51 - 06/25/2013 10:10 AM - Greg Shah

The code in 0621a is good. The findings are also interesting. Please make sure to test cases where "" (empty string) is passed for various parameters AND where unknown value is passed for various parameters. Generally, each test should only pass "" or unknown in one parameter and then normal input for the other parameters. Also note that you will probably have to pass unknown value like this:

```
def var the-unknown-var as char init ?.
get-key-value section the-unknown-var key "some-key" value chVar
```

This is often needed because the 4GL compiler will usually disallow the direct use of the unknown value literal ?.

Good stuff. Keep going.

#52 - 06/26/2013 02:49 PM - Eugenie Lyzenko

- File evl_upd20130626a.zip added

- File testcases_environments_20130626a.zip added

> Please make sure to test cases where "" (empty string) is passed for various parameters AND where unknown value is passed for various parameters. Generally, each test should only pass "" or unknown in one parameter and then normal input for the other parameters. Also note that you will probably have to pass unknown value like this:
>
> ```
> def var the-unknown-var as char init ?.
> get-key-value section the-unknown-var key "some-key" value chVar
> ```
>
> This is often needed because the 4GL compiler will usually disallow the direct use of the unknown value literal ?.

OK. I'll check this. For now the next drop includes some rework for StanzaIni according to recent findings. The changes:

1. Fixed the errors for inclusive substring extracting.
2. The class EnvironmentOps was changed in handling load(character|String,character|String) - the case when we have only two incoming text parameters for load() call. This should mean using load(environmentName, baseKey). The previous case load(environmentName, directory) was wrong because option directory is valid only for INI files, not for registry usage. But INI file usage can only be declared by base-key "INI" option so base-key has priority over directory in case of usage two text options in load() call.
3. StanzaIni was reworked to use String instead of character internal types for Maps elements construction. The reason - the character usage will require double transformation in setKeyValue, getKeyValue calls to trim spaces from character-String-character. And also I consider StanzaIni as some kind of "low level" interface class between P2J with more abstract characters and file system itself. StanzaIni takes characters and returns characters but working internally with Strings. Please let me know if this approach is not acceptable.
4. The HashMap was replaced with LinkedHashMap to preserve the data order taken from INI file or application. Need more tests to be sure LinkedHashMap does the work as expected, HashMap does not for sure(we have chaotic order in resulting INI file).
5. I have disabled the empty line generation. The 4GL does not put extra line between sections. But it can leave such spaces if they were in INI file initially(made manually in plain text editor), need to check this.
6. The testcase set was refreshed. So from now for the INI files too we have some level of debugged working code in Windows.

#53 - 06/26/2013 05:08 PM - Greg Shah

Code Review 0626a

I am fine with the code and approach in general. Please note that the StanzaIni constructors need javadoc. I also do prefer for the EnvironmentAccessor interface (and thus also the EnvironmentDaemon, Registry and StanzaIni classes) be implemented using String instead of character. If there is some behavior that requires character to be used instead, then continue using it. Otherwise, do shift to String. That should also mean that the "unwrapping" and unknown processing can be done in 1 place (EnvironmentOps) which is cleaner.

How much more effort do you expect is needed to complete your testcases and the implementation?

#54 - 06/26/2013 07:21 PM - Eugenie Lyzenko

> How much more effort do you expect is needed to complete your testcases and the implementation?

This is the serious question. I'm trying to work in parallel on INI and registry sections(to be able to think on background task while implementing another) to complete this ASAP. But I've found we need to cover many possible situations, many options, create testcases to see what is happening on real 4GL system and more important to understand why it is happening. So I think at least 2 weeks more to complete is more or less real estimate. Sorry if this is too long. The environments is the point when the error can potentially damage the production system destroying the INI files or registry and I would like to do my best in this task.

#55 - 06/26/2013 07:52 PM - Eugenie Lyzenko

- File evl_upd20130626b.zip added

> Please note that the StanzaIni constructors need javadoc.

The new update uploaded. The javadoc missing entries was added to every java file.

Also I have implemented the approach to preserve possible empty lines between sections in INI file to load. Note it is possible to have more than one empty lines, each one can contain either "" or arbitrary number of spaces or other meaningless characters. For every such a line in one section the unique name is generating. That's because we can not store two different values with same name in Map simultaneously.

#56 - 06/27/2013 10:01 AM - Greg Shah

The code looks good. Some thoughts:

1. Please do consider if it is possible to shift the EnvironmentAccessor to use String instead of character.

2. Does the unrecognized line processing also handle comments?

3. Does the unrecognized line processing even handle comments/whitespace at the beginning of the file (which is not part of any section)?

4. Can there be whitespace or comments in the middle of a section (with real/working content following the whitespace/comments)?

5. I think it may make sense to keep the comments/whitespace separate from the map of maps of key/value. The current approach is good for key/value usage but it is awkward for comments and whitespace. I would prefer you to add some classes:

Content (abstract base class)
LineContent (child of Content which contains a string with any whitespace/comments)
SectionContent (child of Content which contains the section name and a reference to the given section's LinkedHashMap of key/value content)

Then you can maintain a LinkedList<Content> which will handle all the ordering issues and can keep the LineContent separate from the SectionContent.

This approach would need some modification if there can be whitespace/comments in the middle of the section content... Anyway, this may give you some implementation ideas to make a cleaner (and more modular) approach.

#57 - 06/28/2013 06:46 PM - Eugenie Lyzenko

- File testcases_environments_20130628a.zip added

The investigation results:

> 1. Please do consider if it is possible to shift the EnvironmentAccessor to use String instead of character.

Done. I have reworked the code to use String instead of character and boolean instead of logical. Will be uploaded in the next drop.

> 2. Does the unrecognized line processing also handle comments?

Yes but with only exception. If '=' symbol located inside comment - the current implementation consider this line as key=value pair. But if we fix this issue - the comment will be handled OK.

I have made special test(env_test22.p) to check handling of the comment in 4GL INI file. The results are - the comments can be everywhere: at the beginning of the file, in the section start, section end or section middle. The comment sign ';' can even be not the first char in the line(' ;this is also comment line'), the comment can include key-value separator(';this is= also comment line')

> 5. I think it may make sense to keep the comments/whitespace separate from the map of maps of key/value...

OK I'll prepare suggested modifications. On the other hand if we fix the = inside comment issue the current approach will also work. However I agree your suggest is more modular and cleaner.

Another findings. The testcase for unknown-vars(env_test21.p) give very strange results:
1. ? and "" are equivalents.
2. All GET-KEY-VALUE calls with either section or key as unknown just give unknown value on return. There is no list of the section or keys as expected.
3. PUT-KEY-VALUE removes the value when pass unknown value but not as I would expected:

```
...
key=value
...
```

PUT-KEY-VALUE key "key" value "". gives:

```
...
key=
...
```

According to the 4GL docs this case should remove key=value pair from INI file.
If key unknown in PUT-KEY-VALUE statement - 4GL does nothing except adding

```
...
=
...
```

line into INI file. And this is really strange. Because according to 4GL doc this should remove whole section instead.

The summary of the action according to the Progress documentation:
GET-KEY-VALUE statement: section and key-name can be "" or ? - returns the list of the sections or keys.
PUT-KEY-VALUE statement: key and value can be "" or ? - removes the particular key-value pair or whole section(key ?). If section is "" or ? - creating just under current env the key=value pair, this is only for registry based environments.

So the question is which way we are going to go in this point? Will we implement the expected facility here or we will simulate possible 4GL bug?

#58 - 06/30/2013 11:40 AM - Greg Shah

Please implement this exactly like the 4GL actually works. It doesn't matter what the documentation says. We have to make real applications work 100% compatibly with the original. That means we implement all the undocumented and crazy behavior.

#59 - 07/01/2013 07:02 PM - Eugenie Lyzenko

- File evl_upd20130701a.zip added

> ...That means we implement all the undocumented and crazy behavior.

OK. Understood.

The new drop has been uploaded for you to review.
1. The classes except EnvironmentOps has been overwritten to use String instead of character data type. And boolean instead of logical.
2. The INI file logic was changed to use:
Content, LineContent, SectionContent concept.

Start testing and debugging, I've sent the code for you to be able to see on early stage. Continue working, note the comment lines and empty lines are the part of the section now(if it is true for INI file). Looking for the better solution.

#60 - 07/02/2013 10:09 AM - Greg Shah

Code Review 0701a

1. I would like some changes made to the Content class and its subsclasses.

- Do not store "name" in the Content class. This variable only makes sense for a key/value pair and is not a good description for line text.
- You should not need the "withName()" method (see below).
- Please rename the method writeToFile() to write().
- LineContent can store its own data as an instance member called "text".

2. The algorithm for StanzaIni.getContent() is more complicated and slower than needed. Please add a HashMap<String, Content> instance member to StanzaIni. You would add SectionContent instances to that map when they are created and you never have to add LineContent instances to it. Then getContent() becomes a simple map lookup and you no longer need Content.withName().

3. SectionContent should store its own key/value data in a map BUT I really don't want to store the comments and empty lines in the same map. It is messy and confusing. I think it is better to use some simple inner classes and a new list:

```
List<Renderable> ordering = new LinkedList<Renderable>;

public interface Renderable
{
   public String render();
}

private static class TextRenderer
implements Renderable
{
   private String text = null;

   private TextRenderer(String text)
   {
      this.text = text;
   }

   public String render()
   {
      return text;
   }
}

private static class KeyRenderer
implements Renderable
{
   private String key = null;

   private Map<String, String> lookup = null;

   private KeyRenderer(Map<String, String> lookup, String key)
   {
      this.lookup = lookup;
      this.key    = key;
   }

   public String render()
   {
      // code to lookup the value from the key/value map and then output it as a formatted string ("key = valu
e")
   }
}
```

This would easily allow the key/value stuff to be inserted in the list as KeyRenderer instances and all whitespace/comments to be added as TextRenderer instances. The whitespace/comments would no longer be kept in the section key/value map. It should make output easy too.

4. What happens if the same section name appears more than once (2, 3 or more times) in the same stanza INI file?

#61 - 07/02/2013 10:56 AM - Eugenie Lyzenko

> 2. The algorithm for StanzaIni.getContent() is more complicated and slower than needed. Please add a HashMap<String, Content> instance member to StanzaIni. You would add SectionContent instances to that map when they are created and you never have to add LineContent instances to it. Then getContent() becomes a simple map lookup and you no longer need Content.withName().

This means we have to support two collections in StanzaIni class:
1. One is LinkedList<Content> to keep ordering of the lines in the file
2. Another one is HashMap<String, Content> to handle name->section mapping.
Both of them will have the same <Content> entries. Is my understanding correct?

#62 - 07/02/2013 10:57 AM - Greg Shah

Yes.

#63 - 07/02/2013 05:45 PM - Eugenie Lyzenko

- File evl_upd20130702a.zip added

The new drop contains reworked approach for storing key-value pair and ignored lines separately in the data section. Was tested for the comments in start, end and middle of the section. Also checked to keep line ordering for PUT-KEY-VALUE statement.

> 4. What happens if the same section name appears more than once (2, 3 or more times) in the same stanza INI file?

Will check soon. Continue working.

#64 - 07/02/2013 07:30 PM - Eugenie Lyzenko

- File multi_section_test_20130702a.zip added

> 4. What happens if the same section name appears more than once (2, 3 or more times) in the same stanza INI file?

There is interesting result for multisection test. I have attached the test and result here. The evl_env23initial.ini - file before testing, evl_env23.ini - after env_test23.p. The conclusions:
1. If several sections with same names are in the file - only first one is considered as having data.
2. The other sections with this name are completely ignored. Key=value are not reading from 2,3... sections and not writing to them.
3. However these "ghost" sections are not deleted when new data is written to the file, they are considered as "ignored" lines.

#65 - 07/02/2013 08:01 PM - Eugenie Lyzenko

Additional findings for same section name in one file. Even we have the following:

```
...
[Section1]
key1=value1
[Section1]
key2=value2
...
```

when the section are side by side, the key2 become invisible and ignored.

#66 - 07/03/2013 12:54 PM - Greg Shah

Code Review 0702a

The code looks good. Some comments:

1. Please put a TODO into StanzaIni to implement the PrintWriter creation the using proper charset (not always just with the default for the JVM).

2. Have you checked to see what happens if the same key appears more than once in a single section?

#67 - 07/03/2013 01:24 PM - Eugenie Lyzenko

> 1. Please put a TODO into StanzaIni to implement the PrintWriter creation the using proper charset (not always just with the default for the JVM).

OK.

> 2. Have you checked to see what happens if the same key appears more than once in a single section?

In this case only first occurrence line with the duplicated key has meaning. The other lines are silently ignored.

#68 - 07/03/2013 02:38 PM - Greg Shah

> 2. Have you checked to see what happens if the same key appears more than once in a single section?
>
> > In this case only first occurrence line with the duplicated key has meaning. The other lines are silently ignored.

I think that loadIniFile() will need changes to process this properly.

You may need to check if the duplicate keys are deleted on output or if they are just silently left behind.

#69 - 07/03/2013 05:07 PM - Eugenie Lyzenko

- File evl_upd20130703a.zip added

> I think that loadIniFile() will need changes to process this properly.
>
> You may need to check if the duplicate keys are deleted on output or if they are just silently left behind.

OK I'll put a note on this. The today drop handles the key duplications considering them as lines to ignore. Continue working on section duplication case. I'm planning to handle this case as set of the ignored lines not belonging to any sections like comment in the head of the file if you do not mind.

#70 - 07/03/2013 05:09 PM - Greg Shah

> I'm planning to handle this case as set of the ignored lines not belonging to any sections like comment in the head of the file if you do not mind.

OK.

#71 - 07/03/2013 08:06 PM - Eugenie Lyzenko

- File evl_upd20130703b.zip added

The next drop supports the section duplication. Also I have noted one thing. 4GL adds the new key in section after last key=value pair, not exactly at the end of the section(for example if the section ends with empty line or comment). So we have to track the last key=value position(the next to add the new key to be exact). The respective SectionContent member has been added to handle this.

Also I found accidentally one strange thing. The Graphical 4GL interface behaves differently comparing to character one. I mean looks like Graphical UI keep documentation when using "" or ? as key value. And this is more crazy than crazy way we have found before. So may be we will need the code to detect the type of the client at run time.

#72 - 07/04/2013 03:43 PM - Eugenie Lyzenko

Another interesting finding. Consider the situation when there are two keys with same name and different values in one section:

```
...
CharKey2=CharVar2
CharKey2=CharVar3
...
```

Normally when file loading only first one is taken into account. You can get the value, put the value - everything happens for first line. But if we remove the key - the first line will be removed and the second line becomes active, we will have the CharKey2 with value CharVar3. This means after deleting the key we have to reload the INI file to refresh the internal maps or rescan ignored lines for possible key(or section if we had several sections with same name and deleted one) duplications.

#73 - 07/04/2013 05:46 PM - Eugenie Lyzenko

- File testcases_environments_20130704a.zip added

More testcases have been added.

#74 - 07/05/2013 06:36 PM - Eugenie Lyzenko

- File evl_upd20130705a.zip added

The next drop implements the empty or unknown values as parameters for USE, GET-KEY-VALUE and PUT-KEY-VALUE for you to review. The code is under debugging while you reviewing. I have put the call stub to check the interface used(CHUI or not) on client side. Currently the call always returns CHUI so we need to implement more flexible if we still need to differentiate the real behavior of 4GL in these two cases.

Also I have considered the cases of key and section removing when the INI file has duplication for them. The INI file will be reloaded and the new key or section will become active for this case.

The character values "" and unknown are considered to be the same in this drop.

The unload method in StanzaIni now frees the internal map memory.

And I have tested the code for DIR option compatibility and found it is working.

#75 - 07/08/2013 04:20 PM - Eugenie Lyzenko

- File evl_upd20130708a.zip added

- File testcases_environments_20130708a.zip added

The drop includes a bug fix for section remove code and testcases to check. I had to simulate windowing interface while debugging by setting isChui to false because ThinClient.isChui() returns true every time. Tested for working with empty "" parameters for GET and PUT values(Windowing interface type).

Continue working.

#76 - 07/08/2013 06:55 PM - Eugenie Lyzenko

- File evl_upd20130708b.zip added

This drop adds simulation for 4GL CHUI behavior for empty parameters handling. If the value is empty, the key=value pair gets empty value. If key is empty the line

```
=
```

adds after last current key-value pair of the appropriate section.

#77 - 07/09/2013 06:26 PM - Greg Shah

Code Review 0708b

I am fine with the changes. One thing I see is that Javadoc which states If the parameter is null string("") the is confusing since a String instance can be EITHER null OR the empty string "". Please change that text to state empty string ("").

We need to get this task completed. What is your current estimate of the remaining time?

#78 - 07/09/2013 06:53 PM - Eugenie Lyzenko

- File evl_upd20130709a.zip added

This is the fix for issue in EnvironmentOps code for unwrapping character unknown variable. It is possible the code:

```
...
put-key-value section "Section2" key "CharKey3" value the-unknown-var.
...
```

converted as:

```
...
EnvironmentOps.setKeyValue("Section2", "CharKey3", (theUnknownVar).toStringMessage());
...
```

This means passing "?" String to setKeyValue(,,var) confusing StanzaIni and forcing to include detection code to StanzaIni class. Because we decided to make all unwrapping inside EnvironmentOps - appropriate change was made in this update.

> ...Please change that text to state empty string ("").

Fixed in this update too.

> We need to get this task completed. What is your current estimate of the remaining time?

I'm shifting to registry part and I'm expecting to finish it at the end of this week(Anyway ASAP).

#79 - 07/09/2013 09:28 PM - Eugenie Lyzenko

- File evl_upd20130709b.zip added

The special empty parameters cases:

GET-KEY-VALUE
section == "", key != "" -> getting the value for given value name 'key' under current environment
section "", key "" -> getting the comma separated list of all subkey@name pair for current environment
section != "", key "" -> getting the comma separated list of value names for subkey 'section'
section != "", key null -> getting the DEFAULT value for the given subkey 'section'

PUT-KEY-VALUE
section == "", key != "", value != "" -> adding directly under current environment value name 'key' with given 'value'
section != "", key != "", value "" -> deletes name and it's value from subkey 'section'
section != "", key "", value "" -> deletes subkey 'section' with all content
section != "", key "", value != "" -> sets new value for DEFAULT value name for subkey 'section'

The next drop includes change for Registry.java preparation for remaining native calls. The next step - implement remaining declared JNI.

#80 - 07/10/2013 08:51 PM - Eugenie Lyzenko

- File evl_upd20130710a.zip added

This drop includes JNI implementation for Windows registry access functions and minor change for JNI signature. Debugging and testing will be started tomorrow while you are able to review the logic approach for possible notes.

#81 - 07/11/2013 06:58 AM - Greg Shah

Code Review 0710a

1. The Registry.java class javadoc needs significant enhancements. The native methods have only a 1-line description. It does not really tell anything about the full details of how those methods work. Even the non-native methods need much more documentation. Every parameter needs to be completely documented, especially since most of the complexity of the behavior comes from different combinations of parameter edge-cases.

2. As a general rule, all of the files are lacking complete javadoc about what happens when a parameter is null or unknown value. Some of the parameters are missing documentation on what happens when the parameter is an empty string.

3. There are many places in the code that are structured like this:

```
      if ((section == null || section.isEmpty()) && (key != null && !key.isEmpty()))
      {
         ...
      }
      else if ((section == null || section.isEmpty()) && (key == null || key.isEmpty()))
      {
         ...
      }
      else if ((section != null && !section.isEmpty()) && (key != null && key.isEmpty()))
      {
         ...
      }
      else if ((section != null && !section.isEmpty()) && key == null)
      {
         ...
      }
      else
      {
         ...
      }
```

This structure is very hard to read (especially when you add the hidden code back in. The duplication of condition processing makes the reader have to carefully evaluate what is going on and the conditions are so complex that it takes much longer than it should to do that analysis. This will make the code hard to maintain and fragile to regressions and bugs. Please look through this update for such cases and replace them with an approach like this:

```
      if (section == null || section.isEmpty())
      {
         if (key != null && !key.isEmpty())
         {
            ...
         }
         else
         {
            ...
         }
      }
      else
      {
         if (key == null)
         {
            ...
         }
         else if (key.isEmpty())
         {
            ...
         }
         else
         {
            ...
         }
      }
```

Please take this approach in all of your code in the future. Let me know if the reason for this unclear.

#82 - 07/11/2013 08:55 AM - Eugenie Lyzenko

- File evl_upd20130711a.zip added

1, 2.

OK, agreed, we need more descriptions here. I'll add detailed javadocs in recent today drop.

3.

The point #3 reworked. Let me know if this is acceptable now or not.

#83 - 07/11/2013 12:02 PM - Eugenie Lyzenko

Finding news for handling registry in Windows 4GL in PUT/GET KEY-VALUE calls.
1. The CHUI and GUI interfaces works the same.
2. The "" and unknown-var considered the same way.
3. The crazy news has too. For both interface types the behavior differs from what is described in Progress documentation:
The case when section "" and key "" GET-KEY-VALUE should return the comma separated section - key pairs in the form:
"section1@key1,section1@key2,...,sectonN@keyN"
But actually this case returns just comma separated list of the sections in format "section1,section2,...,sectionN".

So I need to make the appropriate changes in native code to take into account #3.

#84 - 07/11/2013 01:33 PM - Greg Shah

OK.

#85 - 07/11/2013 07:39 PM - Eugenie Lyzenko

- File evl_upd20130711b.zip added

- File testcases_environments_20130711a.zip added

The second drop for today includes Java doc additions for Registry class. Please let me know if this level of details is enough or not.

Also the new testcases have been added to check registry functionality. The native code has small fix to use Windows 2000+ compatible call to remove registry entry and rework for getting key list function. Looks like the native part works OK. I'll think about more complex test but the edge conditions handling the same as for real 4GL. I'm a bit worry about detecting CHUI and GUI mode for StanzaIni calls, can we use some native call located in registry.c to get this option for current client process instead of ThinClient.isChui()?

One more note: the debugging printf() calls are still in registry.c.

#86 - 07/12/2013 11:56 AM - Eugenie Lyzenko

- File testcases_environments_20130712a.zip added

- File evl_upd20130712a.zip added

The drop includes:
1. Disabling environment related calls and daemon starting for non-Windows environment.
2. Adding handling of two missed edge cases for work with DEFAULT registry value name and key without a section located just under current environment. The rules are:
section "", key != "", value "" -> removing directly under current environment value name 'key' with its 'value'
section != "", key null, value "" -> sets DEFAULT value name with unset for subkey 'section'
3. The testcase env_test32.p has been added to check DEFAULT key special case.

The debugging code in registry.c is still not removed.

So I'm expecting to finish working on this task today.

#87 - 07/12/2013 05:36 PM - Eugenie Lyzenko

- File evl_upd20130712b.zip added

This is the final changes for registry access for you to review. Changes comparing to previous update:
1. The native c code cleaned from debugging calls.
2. The no-error option support has been included into java code, EnvironmentOps.java source.

The only opened point for now I can tell is detecting the CHUI or GUI client mode in Windows. But I guess when we implement GUI client for Windows - the ThinClient.isChui() will return proper value so the current approach here is good enough.

Let me know please if you have any notes. I'll be online 1-1.5 hours more for possible urgent changes

#88 - 07/12/2013 07:24 PM - Eugenie Lyzenko

- File evl_upd20130712c.zip added

This is the StanzaIni.java Javadoc adding for more descriptions and little syntax fix for Javadoc in Registry.java.

#89 - 07/17/2013 05:24 PM - Greg Shah

Code Review 0712c

1. Do LOAD/UNLOAD/USE really generate a STOP condition (like a CTRL-C) on non-Windows?

2. Normally, when NO-ERROR is active the ERROR-STATUS handle can be read to get the error message, error number and so forth. This is handled in ErrorManager using the RemoteErrorData.addRecord() method. But showErrorAndAbend() does not implement this feature. Please check if you can see the error data using ERROR-STATUS after you "catch" the STOP condition using an extra containing block like:

```
REPEAT ON STOP UNDO, LEAVE:
  /* code that uses LOAD/UNLOAD/USE on non-Windows platform */

  LEAVE.
END.

/* code that checks ERROR-STATUS */
```

3. Why are there comments like // TODO implement this in these files? Are these really still open TODOs?

4. Are all your testcases working exactly like in the 4GL?

#90 - 07/17/2013 05:44 PM - Eugenie Lyzenko

1. Do LOAD/UNLOAD/USE really generate a STOP condition (like a CTRL-C) on non-Windows?

Yes, when error is enabled these statements restarts the application. If no-error - silently ignoring.

2. Normally, when NO-ERROR is active the ERROR-STATUS handle can be read to get the error message, error number and so forth....

OK. I'll check this.

3. Why are there comments like // TODO implement this in these files? Are these really still open TODOs?

I've left them for possible changes reflecting to your notes. Do I need to remove them?

Note the PrinterWriter TODO is actual because there is only default charset is now works.

4. Are all your testcases working exactly like in the 4GL?

Yes, I've tested single functionality with single test. Then I debugged the code to fix deviations. So the the goal was to have identical results on Windows in 4GL and P2J. And I see no difference for test behavior for now.

#91 - 07/17/2013 05:57 PM - Greg Shah

> I've left them for possible changes reflecting to your notes. Do I need to remove them?

If your testing proves that the implementation is correct and complete, then please remove the TODOs. If something is still missing, then put a more specific TODO in place to make it clear.

> Note the PrinterWriter TODO is actual because there is only default charset is now works.

Yes, good.

>> 4. Are all your testcases working exactly like in the 4GL?

> Yes, I've tested single functionality with single test. Then I debugged the code to fix deviations. So the the goal was to have identical results on Windows in 4GL and P2J. And I see no difference for test behavior for now.

When you say you tested a "single test", do you mean that you ran each of your testcases individually? In other words, it is important that ALL of your testcases are working, not just one of them.

#92 - 07/17/2013 06:07 PM - Eugenie Lyzenko

> If your testing proves that the implementation is correct and complete, then please remove the TODOs. If something is still missing, then put a more specific TODO in place to make it clear.

OK.

> When you say you tested a "single test", do you mean that you ran each of your testcases individually? In other words, it is important that ALL of

> your testcases are working, not just one of them.

To be exact I should say "every single test" instead. Yes I ran each of the test individually. From first to last. Debug, fix the code and create new test and so on. So ALL testcases are working.

#93 - 07/17/2013 08:16 PM - Eugenie Lyzenko

- File testcases_environments_20130717a.zip added

- File evl_upd20130717a.zip added

The update with removed TODO comments for implemented code. The testcases also refreshed.

The testcase env_test34.p shows error-status:error == no on 4GL Linux system. The current EnvironmentOps code also has the same behaviour with usage showErrorAndAbend(). The only issue here is the showErrorAndAbend() has additional internal LogicalTerminal.pause() integrated which 4GL does not do. Can we make the method similar to showErrorAndAbend() but without internal LogicalTerminal.pause()? Because looks like 4GL does not consider INI/Registry code as deserved to produce the real error if encountered in Linux. The application just stops when NO-ERROR is not defined.

#94 - 07/18/2013 11:20 AM - Greg Shah

Code Review 0717a

Each code review I look deeper in some areas and catch some things I didn't see before. We are getting close, but there is some cleanup to do.

1. The EnvironmentAccessor interface is designed well for use as the remote interface to EnvironmentDaemon. But it is not a good fit for the "low level" Register and StanzaIni classes. In particular:

- The use() method is not needed in either the Registry or StanzaIni classes.
- The load() method arguments are not used in either the Registry or StanzaIni classes.
- The unload() method arguments are not used in either the Registry or StanzaIni classes.

Please create an EnvironmentReader interface, that is a simpler/minimized version of EnvironmentAccessor. Change the Registry or StanzaIni classes to implement EnvironmentReader instead of EnvironmentAccessor.

2. In Registry.java, the javadoc has "defailt" instead of "default".

3. There is dead code in Environments.WorkArea: remove the lastError comments.

4. In regard to the error/abend processing:

> The testcase env_test34.p shows error-status:error == no on 4GL Linux system. The current EnvironmentOps code also has the same behaviour with usage showErrorAndAbend(). The only issue here is the showErrorAndAbend() has additional internal LogicalTerminal.pause() integrated which 4GL does not do. Can we make the method similar to showErrorAndAbend() but without internal LogicalTerminal.pause()? Because looks like 4GL does not consider INI/Registry code as deserved to produce the real error if encountered in Linux. The application just stops when NO-ERROR is not defined.

Yes, it makes sense to have a new method in ErrorManager that provides the proper behavior. Please name it conditionalShowErrorAndAbend() and make sure that it handles the NO-ERROR mode too (it should return silently if we are in silent error mode). That way you can further simplify the calling code to avoid the call to ErrorManager.isSilentError().

#95 - 07/18/2013 01:14 PM - Eugenie Lyzenko

- File evl_upd20130718a.zip added

OK. The changes for points 1-4 has been uploaded for review. NO-ERROR mode is also working for ErrorManager change.

#96 - 07/18/2013 01:30 PM - Greg Shah

Code Review 0718a

It looks good. Get it runtime regression tested on devsrv01.

#97 - 07/20/2013 03:31 PM - Eugenie Lyzenko

Finally the update has passed all regression testing in devsrv01 except known TC-JOB-002(including CTRL-C 3-way tests). I'm going to merge it with the recent P2J codebase and upload here.

#98 - 07/22/2013 10:18 AM - Eugenie Lyzenko

- File evl_upd20130722a.zip added

This is the drop merged with recent P2J codebase(10365). Do we need to run runtime-testing again? Or I can check it with bzr?

#99 - 07/25/2013 08:34 AM - Greg Shah

I think the merged changes are safe enough. Go ahead and check it in (and distribute it via email). Good work!

#100 - 07/25/2013 10:27 AM - Eugenie Lyzenko

Committed in bzr as 10367. Will distribute soon.

#101 - 07/31/2013 03:22 PM - Greg Shah

- Assignee set to Eugenie Lyzenko

- Status changed from WIP to Closed

#102 - 11/16/2016 11:43 AM - Greg Shah

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

Files

evl_upd20130128a.zip	13.4 KB	01/29/2013	Eugenie Lyzenko
registry.p.zip	401 Bytes	01/29/2013	Eugenie Lyzenko
registry_20130129a.p.zip	499 Bytes	01/29/2013	Eugenie Lyzenko
evl_upd20130129a.zip	13.7 KB	01/29/2013	Eugenie Lyzenko
evl_upd20130130a.zip	13.9 KB	01/30/2013	Eugenie Lyzenko
evl_upd20130131a.zip	14 KB	01/31/2013	Eugenie Lyzenko
evl_upd20130203a.zip	14 KB	02/03/2013	Eugenie Lyzenko
evl_upd20130605a.zip	17.5 KB	06/05/2013	Eugenie Lyzenko
evl_upd20130610a.zip	129 KB	06/10/2013	Eugenie Lyzenko
evl_upd20130614a.zip	129 KB	06/14/2013	Eugenie Lyzenko
testcases_environments_20130614a.zip	3.15 KB	06/14/2013	Eugenie Lyzenko
evl_upd20130618a.zip	150 KB	06/18/2013	Eugenie Lyzenko
evl_upd20130621a.zip	142 KB	06/21/2013	Eugenie Lyzenko
testcases_environments_20130621a.zip	6.92 KB	06/21/2013	Eugenie Lyzenko
evl_upd20130626a.zip	143 KB	06/26/2013	Eugenie Lyzenko
testcases_environments_20130626a.zip	7.86 KB	06/26/2013	Eugenie Lyzenko

evl_upd20130626b.zip	144 KB	06/26/2013	Eugenie Lyzenko
testcases_environments_20130628a.zip	10.9 KB	06/28/2013	Eugenie Lyzenko
evl_upd20130701a.zip	156 KB	07/01/2013	Eugenie Lyzenko
evl_upd20130702a.zip	156 KB	07/02/2013	Eugenie Lyzenko
multi_section_test_20130702a.zip	1.18 KB	07/02/2013	Eugenie Lyzenko
evl_upd20130703a.zip	148 KB	07/03/2013	Eugenie Lyzenko
evl_upd20130703b.zip	148 KB	07/04/2013	Eugenie Lyzenko
testcases_environments_20130704a.zip	11 KB	07/04/2013	Eugenie Lyzenko
evl_upd20130705a.zip	159 KB	07/05/2013	Eugenie Lyzenko
evl_upd20130708a.zip	159 KB	07/08/2013	Eugenie Lyzenko
testcases_environments_20130708a.zip	13.7 KB	07/08/2013	Eugenie Lyzenko
evl_upd20130708b.zip	159 KB	07/08/2013	Eugenie Lyzenko
evl_upd20130709a.zip	150 KB	07/09/2013	Eugenie Lyzenko
evl_upd20130709b.zip	151 KB	07/10/2013	Eugenie Lyzenko
evl_upd20130710a.zip	161 KB	07/11/2013	Eugenie Lyzenko
evl_upd20130711a.zip	161 KB	07/11/2013	Eugenie Lyzenko
evl_upd20130711b.zip	162 KB	07/11/2013	Eugenie Lyzenko
testcases_environments_20130711a.zip	15.5 KB	07/11/2013	Eugenie Lyzenko
evl_upd20130712a.zip	162 KB	07/12/2013	Eugenie Lyzenko
testcases_environments_20130712a.zip	16.1 KB	07/12/2013	Eugenie Lyzenko
evl_upd20130712b.zip	162 KB	07/12/2013	Eugenie Lyzenko
evl_upd20130712c.zip	162 KB	07/12/2013	Eugenie Lyzenko
evl_upd20130717a.zip	153 KB	07/18/2013	Eugenie Lyzenko
testcases_environments_20130717a.zip	14.7 KB	07/18/2013	Eugenie Lyzenko
evl_upd20130718a.zip	165 KB	07/18/2013	Eugenie Lyzenko
evl_upd20130722a.zip	176 KB	07/22/2013	Eugenie Lyzenko