User Interface - Feature #4517

optionally back the 4GL features for Registry access with the user-specific offline storage features

01/23/2020 05:20 PM - Greg Shah

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

Description

Related issues:

Related to User Interface - Feature #4286: support user-specific offline stor...	**Closed**
Related to User Interface - Feature #3880: enhanced browse 3rd phase of impro...	**WIP**
Related to User Interface - Feature #4854: origin affinity	**Closed**
Related to User Interface - Feature #4129: map web client users to OS accounts	**Closed**
Related to User Interface - Feature #7997: Encryption for client storage data...	**New**
Related to User Interface - Bug #8353: Fully support stanza INI LOAD/USE/UNLO...	**WIP**

History

#1 - 01/23/2020 05:33 PM - Greg Shah

In #4286, an offline storage facility was created that allows persistent storage/retrieval of key/value pairs in the user's local system. This implementation depends upon the client driver being used. For non-web based clients, the Java Preferences.userRoot() is used. For web clients, the browser's local storage API is used. In both cases, the key/value pairs can survive across client restarts. The facility is included in trunk revision 11330. This initially is planned for usage by the enhanced browse (see #3880) to store user-specific configuration choices.

This task in intended to provide a mechanism that can be used at the 4GL code level. Now that we have a functioning direct Java access mechanism (see #3867), we can create a new helper class that provides a simple API for this service. Then 4GL code can access these Java classes directly to store/retrieve key/value pairs. We will document this as an enhancement capability of FWD, but at this time I think there is no reason to actually provide 4GL syntax changes. I just want the helper class to provide a clean and simple API that is easy to use by 4GL code.

One customer is planning to use this to store a kind of "device id" that differentiates the user's system in their application. Other customers may use this to store data that they would previously have stored in an ini file or in the Windows registry. Please start with reviewing the current offline storage capability. Then design an API that meets the requirements above. Post it here for review. The implementation won't be hard.

#2 - 01/23/2020 05:33 PM - Greg Shah

- Related to Feature #4286: support user-specific offline storage that is provided by the client's UI driver added

#3 - 01/23/2020 05:33 PM - Greg Shah

- Related to Feature #3880: enhanced browse 3rd phase of improvements added

#4 - 08/14/2020 08:44 AM - Greg Shah

- Subject changed from create a facility available from converted 4GL code which allows persistent storage/retrieval of user-specific key/value pairs to

optionally back the 4GL features for Registry access with the user-specific offline storage features

The 4GL features LOAD, UNLOAD, USE, GET-KEY-VALUE, PUT-KEY-VALUE provide a mechanism to read and write key/value data from/into either INI files (any platform) or the Registry (Windows only). We support this fully.

Today, on non-Windows platforms, if the 4GL code is trying to access the registry, it will not work. When the FWD client runs on Windows it does work because the registry support is backed by native code (JNI) that handles reading/writing the registry using WIN32.

Nearly all 4GL GUI applications we have ever seen use the registry for some configuration. It seems silly that we don't transparently map this access into a similar feature on non-Windows platforms. This would allow any such code to just work instead of requiring it to be re-written.

In #4286, a persistent offline storage facility was created. This allows the storage and retrieval of key/value pairs from a client-specific storage medium (local storage in the browser and Java-based user preferences in non-web clients). This seems to be a perfect match for backing the registry support.

System-wide values won't be able to shared by multiple users, but that really should not matter much. The storage will be per-user and that is OK.

The other advantage of this approach is that there is no new 4GL syntax needed. We can implement this completely in the runtime.

#5 - 08/14/2020 08:45 AM - Greg Shah

- Related to Feature #4854: origin affinity added

#6 - 08/14/2020 08:45 AM - Greg Shah

- Related to Feature #4129: map web client users to OS accounts added

#7 - 08/14/2020 08:50 AM - Greg Shah

In #4129 we are implementing features to implement a stable mapping of end users to OS accounts. This will be needed for non-web clients to have a stable persistent storage.

In #4854, we are implementing features to implement a stable mapping of browser + user to the FWD client Jetty host + port. This will be needed for web clients to have a stable persistent storage.

#8 - 06/16/2023 05:38 AM - Galya B

What class implements the global 4GL methods accessing the Windows registry store?

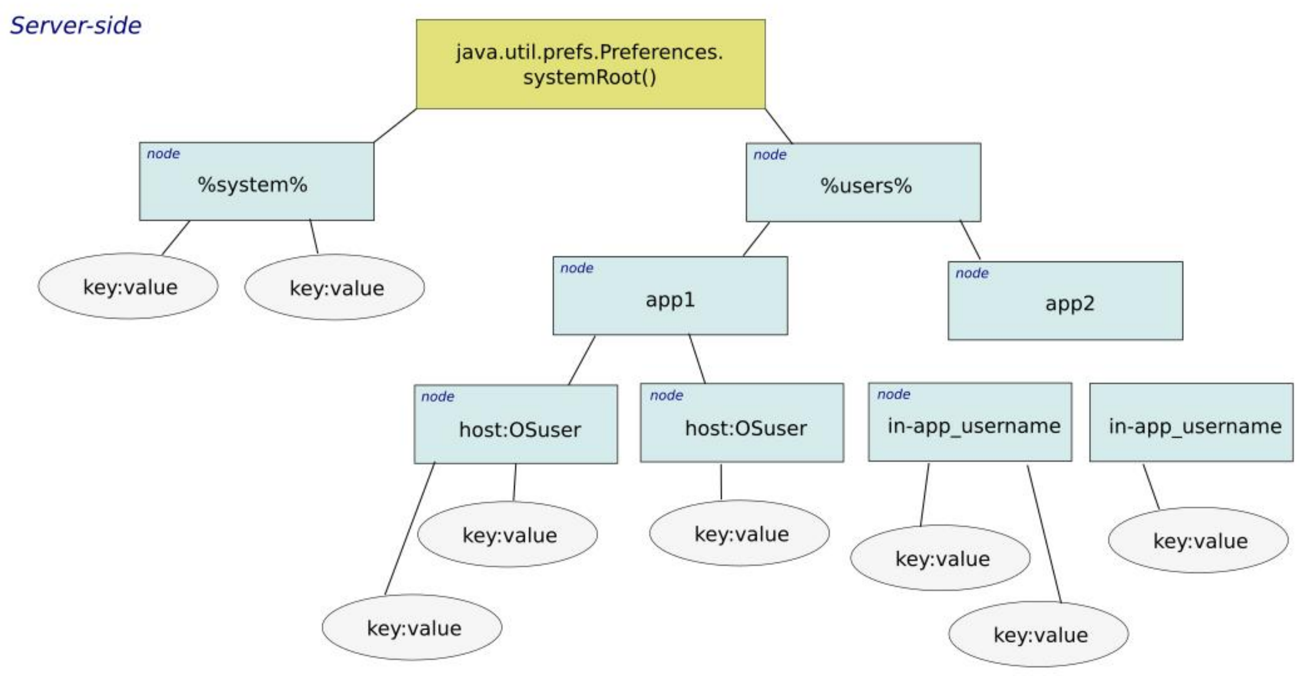
#9 - 06/16/2023 06:24 AM - Galya B

To sum it up: This FWD persistent storage should be used:
1. On non-Windows operating system for global environment storage, alternative to Windows registry, accessible through the global 4GL methods LOAD, UNLOAD, USE, GET-KEY-VALUE, PUT-KEY-VALUE.
2. On all operating system and all types of drivers for storing BROWSE widget enhancement preferences for in-app users (or OS users when no in-app users known). Accessible in client processes through the driver's KeyValueStorageAdapter.

Do we expect multiple different apps to use the enhanced BROWSE on the same host? For example several swing clients running different procedures will use the same OS account, but may need different preferences?

#10 - 06/16/2023 07:22 AM - Galya B

- File preferences.png added



#11 - 06/16/2023 03:26 PM - Greg Shah

Galya B wrote:

> What class implements the global 4GL methods accessing the Windows registry store?

To the converted code, the 4GL calls will be converted into methods in EnvironmentOps. That code downcalls to the client through the normal RemoteObject protocol and the EnvironmentAccessor interface. The client side implementation is in EnvironmentDaemon, it supports both "stanza-based" INI files and (when on Windows) the WIN32 registry. The registry access is via JNI code.

> Do we expect multiple different apps to use the enhanced BROWSE on the same host?

At this time the browse does not consider an "app" level, but rather there are global, user and browse-level settings.

#12 - 10/11/2023 01:47 AM - Galya B

- Status changed from New to WIP

- Assignee set to Galya B

#13 - 10/26/2023 03:19 AM - Galya B

Storage for GET-KEY-VALUE and PUT-KEY-VALUE:

OS	LOAD BASE-KEY "INI"	Client Type	Storage	Storage lifespan	Shared between
Win	▯	Web	Server/Broker machine HKEY_CURRENT_USER	OS reinstall	All end-users with the same OS user on the server/broker machine
Win	▯	Appserver, Scheduled batch	Server/Broker machine HKEY_CURRENT_USER	OS reinstall	All end-users with the same OS user on the server/broker machine
Win	▯	Desktop	Client machine HKEY_CURRENT_USER	OS reinstall	All end-users with the same OS user on the client machine
Win/Other	▯	Web	Server/Broker machine .ini file	Disk change, cleanup	All web users and processes on the server/broker machine
Win/Other	▯	Appserver, Scheduled batch	Server/Broker machine .ini file	Disk change, cleanup	All web users and processes on the server/broker machine
Win/Other	▯	Desktop	Client machine .ini file	Disk change, cleanup	All end-users on the client machine
Other	▯	Web	With webClient/useLocalStorage -> localStorage	Browser storage cleanup	Unique for the in-app user (with SSO) or same for all browser users
			Without webClient/useLocalStorage -> Server/Broker machine JVM Preferences.userRoot	JVM reinstall?, version?	All end-users with the same OS user on the server/broker machine
Other	▯	Appserver, Scheduled batch	Server/Broker machine JVM Preferences.userRoot File in [userhome]/.java/.userPrefs on Linux	JVM reinstall?, version?	All end-users with the same OS user on the server/broker machine
Other	▯	Desktop	Client machine JVM Preferences.userRoot File in [userhome]/.java/.userPrefs on Linux	JVM reinstall?, version?	All end-users with the same OS user on the client machine

> One customer is planning to use this to store a kind of "device id" that differentiates the user's system in their application. Other customers may use this to store data that they would previously have stored in an ini file or in the Windows registry.

There is some inconsistency in the level of collision between users who will share the same storage in the different cases, but at the moment I can't make something out of it.

#14 - 10/26/2023 04:43 AM - Galya B

The most obvious problem I see right away is with web users on Windows. All of them would share this "device id" the customer is planning to store. Anything "user" specific will be shared with all other web users, if the webClient/defaultOsUser is configured. Even worse with .ini on any OS is how all processes, no matter what OS user they are running under, can potentially share the same file. One of the differences in FWD in comparison to 4GL is that "user" is no longer an OS user on that same machine.

#15 - 10/27/2023 09:11 AM - Galya B

Do we want all web clients always to store registry in the browser for consistency?

#16 - 10/27/2023 09:11 AM - Galya B

The client process for web is on the server if no brokers are configured. And brokers are reused by multiple clients.

#17 - 10/27/2023 09:13 AM - Greg Shah

> Do we want all web clients always to store registry in the browser for consistency?

No, some customers will want server-based storage. But I do think that the registry APIs should always be backed by some non-registry storage when not running on Windows.

#18 - 10/27/2023 09:16 AM - Greg Shah

The table in #4517-13 is about FWD behavior (I think). Is this meant to document the current behavior before any changes?

#19 - 10/27/2023 09:16 AM - Galya B

Do we want the web under linux to also store in Preferences.userRoot? It will be more consistent to everything else, where the storage is OS user dependent.

#20 - 10/27/2023 09:17 AM - Galya B

Greg Shah wrote:

> The table in #4517-13 is about FWD behavior (I think). Is this meant to document the current behavior before any changes?

The last three rows are new behavior by substituting non-Win, non-ini registry with user storage.

#21 - 10/27/2023 09:54 AM - Galya B

- % Done changed from 0 to 90

I'm almost ready with the implementation. Can I have a review of the table?

#22 - 10/27/2023 10:08 AM - Greg Shah

Review of #4517-13:

- Shouldn't the "Shared between" column for the Win/other scenarios that use LOAD BASE-KEY "INI" report "Anyone with file system access to that .ini file."?
- For OS "Other" can you please clarify the implementation idea for "Unique for the in-app user (with SSO) or same for all browser users"?

  Do we want the web under linux to also store in Preferences.userRoot? It will be more consistent to everything else, where the storage is OS user dependent.

It is a good question. I generally like it and some customers will certainly find it very useful. However, it won't work for customers that want to store a "device-id" or who want different behavior based on which browser instance is being used.

What is the effort to implement it both ways and allow configuration of the approach to use?

I can hear the groans now. Sorry. :)

#23 - 10/27/2023 10:15 AM - Galya B

Greg Shah wrote:

> Review of #4517-13:
>
> > Do we want the web under linux to also store in Preferences.userRoot? It will be more consistent to everything else, where the storage is OS user dependent.
>
> It is a good question. I generally like it and some customers will certainly find it very useful. However, it won't work for customers that want to store a "device-id" or who want different behavior based on which browser instance is being used.
>
> What is the effort to implement it both ways and allow configuration of the approach to use?
>
> I can hear the groans now. Sorry. :)

Not this time, because I raised the question imagining the consequences. I'm already done with the implementation as it is in the table, so adding a secondary alternative may require new methods exposed by the ClientExport. It's doable, but then can we make localStorage the non-default, because it's more different?

> - Shouldn't the "Shared between" column for the Win/other scenarios that use LOAD BASE-KEY "INI" report "Anyone with file system access to that .ini file."?

Yes, there are other details that could have been added to that column like "Disk failure", but I decided to keep it simple. The thing is that some customers use defaultOsUser, so all their web users will have the same access.

> - For OS "Other" can you please clarify the implementation idea for "Unique for the in-app user (with SSO) or same for all browser users"?

Well, localStorage being used for browse configs initially, that are very personal led to the introduction of storageId somewhere in #3931 (I'm pretty sure I mentioned it) that allows the same localStorage to be store in independent keys for each in-app user. Directly reusing web storage as it is for registry will cause the effect described in the table.

#24 - 10/27/2023 10:27 AM - Greg Shah

> It's doable, but then can we make localStorage the non-default, because it's more different?

Yes, that makes sense.

#25 - 10/30/2023 11:28 AM - Galya B

Browse configs will continue to be saved only in localStorage, while registry will fallback to Preferences by default and will have to be configured explicitly to use localStorag. Right? Or both act the same way?

#26 - 10/30/2023 03:34 PM - Greg Shah

I'd prefer if both act the same way (default to preferences).

#27 - 11/01/2023 03:21 AM - Galya B

Why is WebSocket messaging using only byte arrays and no text messages? Is there a reason behind that decision or it was purely for consistency allowing binary data to pass through?

#28 - 11/01/2023 07:46 AM - Greg Shah

It is for consistency of the protocol and for performance.

#29 - 11/01/2023 11:27 AM - Galya B

- Status changed from WIP to Review

- % Done changed from 90 to 100

4854a r14802 ready for review. Based on trunk r14792. Changes solve #4854, #4517 and a few unrelated small issues, including #7956-4, #3931-464. Be open-minded with the review.

#30 - 11/02/2023 10:19 AM - Galya B

- Related to Feature #7997: Encryption for client storage data / registry added

#31 - 11/07/2023 06:55 AM - Galya B

- Status changed from Review to Test

4854a was merged to trunk as rev. 14823 and archived.

It's important customers to get familiar with the table in #4517-13 and decide to enable webClient/useLocalStorage or not.

#32 - 02/07/2024 01:56 PM - Greg Shah

- Status changed from Test to Closed

#33 - 02/07/2024 04:59 PM - Roger Borrello

There is a regression caused by revision 14823 to src/com/goldencode/p2j/util/EnvironmentDaemon.java method public String use(String env).

The code had:

```
         else
         {
            currentEnv = envIniMap.get(env);
            if (currentEnv != null)
            {
               return currentEnv.getEnvironmentName();
            }

            // not found
            env = null;
         }
```

Which was expanded to extra cases:

```
...
      // stanza third
      currentEnv = envIniMap.get(envIni);
      if (currentEnv != null)
      {
         return currentEnv.getEnvironmentName();
      }

      // fallback
      currentEnv = envFallbackMap.get(env);
      if (currentEnv != null)
      {
         return currentEnv.getEnvironmentName();
      }

      // not found
      return null;
...
```

Because the envIni was used to lookup the environment (instead of env), it isn't being found. By modifying to:

```
...
      // stanza third
      currentEnv = envIniMap.get(env);
...
```

We find the environment.

In my research, I came across a regression that has been around in src/com/goldencode/p2j/util/StanzaIni.java since revision 11266 where the environment name is being stored with **.ini** appended to the end. This came about when the constructor was changed from:

```
   public StanzaIni(String env, String directory, boolean flagNew, boolean readOnly)
   {
      this();

      name = env.endsWith(".ini") ? env : env + ".ini";
      if (directory != null)
      {
         workingDirectory = directory;
      }
      isNew = flagNew;

      isReadOnly = readOnly;

      // environment name should not contain .ini file extension
      envName = env;
   }
```

to modifying env:

```
   public StanzaIni(String env, String directory, boolean flagNew, boolean readOnly)
   {
      contList = new LinkedList<Content>();
      sectMap = new TreeMap<String, Content>(String.CASE_INSENSITIVE_ORDER);

      // get driver type and read only property.
      ThinClient tc = ThinClient.getInstance();
      isChui = tc.isChui();

      if (env.indexOf('.') < 0)
      {
         env = env + ".ini";
      }

      name = env;
      if (directory != null)
      {
         workingDirectory = directory;
      }
      isNew = flagNew;

      isReadOnly = readOnly;

      // environment name should not contain .ini file extension
      envName = env;
   }
```

This can be easily fixed:

```
--- /tmp/brz-diff-c2sl6_b2/old/src/com/goldencode/p2j/util/StanzaIni.java
+++ /home/rfb/projects/fwd/trunk_14929-bad/src/com/goldencode/p2j/util/StanzaIni.java
@@ -154,6 +156,9 @@
       // get driver type and read only property.
       ThinClient tc = ThinClient.getInstance();
       isChui = tc.isChui();
+
+      // environment name should not contain .ini file extension
+      envName = env;

       if (env.indexOf('.') < 0)
       {
@@ -168,9 +173,6 @@
       isNew = flagNew;

       isReadOnly = readOnly;
-
-      // environment name should not contain .ini file extension
-      envName = env;
    }

    /**
```

Can I create a branch here for the updates?

#34 - 02/07/2024 06:05 PM - Greg Shah

Yes, you can create a branch. The proposed patch doesn't seem quite right. Shouldn't we be testing to see if env already ends in .ini and removing it if it is there before assigning it to envName?

#35 - 02/08/2024 01:58 AM - Galya B

Roger Borrello wrote:

> There is a regression caused by revision 14823 to src/com/goldencode/p2j/util/EnvironmentDaemon.java method public String use(String env).
>
> The code had:
> [...]
> Which was expanded to extra cases:
> [...]
>
> Because the envIni was used to lookup the environment (instead of env), it isn't being found. By modifying to:
> [...]
> We find the environment.

Roger, this is not so.

Current version of EnvironmentDaemon.use:

```
   public String use(String env)
   {
      // Empty string as environment makes the default environment the current one
      if (env == null || env.isEmpty())
      {
         currentEnv = null;
         return "";
      }

      // registry first
      currentEnv = envRegMap.get(env.toLowerCase());
      if (currentEnv != null)
      {
         return currentEnv.getEnvironmentName();
      }

      String envIni = env.indexOf('.') >= 0 ? env : env + ".ini";

      // default ini second
      if (defaultEnv != null && envIni.equals(defaultEnv.getEnvironmentName()))
      {
         currentEnv = null;
         return "";
      }

      // stanza third
      currentEnv = envIniMap.get(envIni);
      if (currentEnv != null)
      {
         return currentEnv.getEnvironmentName();
      }

      // fallback
      currentEnv = envFallbackMap.get(env.toLowerCase());
      if (currentEnv != null)
      {
         return currentEnv.getEnvironmentName();
      }

      // not found
      return null;
   }
```

The previous version of EnvironmentDaemon.use:

```
   public String use(String env)
   {
      // Empty string as environment makes the default environment the current one
      if (env == null || env.isEmpty())
      {
         currentEnv = null;
         env = "";
      }
      else
      {
         // registry first
         currentEnv = envRegMap.get(env.toLowerCase());
         if (currentEnv != null)
         {
            return currentEnv.getEnvironmentName();
         }

         // stanza second
         if (env.indexOf('.') < 0)
         {
            env = env + ".ini";
         }

         if (defaultEnv != null && env.equals(defaultEnv.getEnvironmentName()))
         {
            // switch to default environment loaded from ini
            currentEnv = null;
            env = "";
         }
         else
         {
            currentEnv = envIniMap.get(env);
            if (currentEnv != null)
            {
               return currentEnv.getEnvironmentName();
            }

            // not found
            env = null;
         }
      }

      return env;
   }
```

You might get confused by the original code variable being reassigned, but in actual fact it was env ini at the point of searching through the envIniMap. Do you see this line in the original code: env = env + ".ini";?

#36 - 02/08/2024 08:57 AM - Roger Borrello

Galya B wrote:

> You might get confused by the original code variable being reassigned, but in actual fact it was env ini at the point of searching through the envIniMap. Do you see this line in the original code: env = env + ".ini";?

All I know is what I see... No crash when opening the color palette in the customer application. I'll create a branch so I can more fully test.

With respect to branch... Galya, your #4517-31 post shows branch **4854a** merged and archived, and I see branch **4517a** is active. Was that just a mistake when archiving?

> 4854a was merged to trunk as rev. 14823 and archived.

#37 - 02/08/2024 09:02 AM - Galya B

Roger Borrello wrote:

> Galya B wrote:
>
> > You might get confused by the original code variable being reassigned, but in actual fact it was env ini at the point of searching through the envIniMap. Do you see this line in the original code: env = env + ".ini";?
>
> All I know is what I see...

Since you're seeing, then we're on the same opinion that there is no regression in r14823. Go ahead and fix the r11266 regression.

> With respect to branch... Galya, your #4517-31 post shows branch **4854a** merged and archived, and I see branch **4517a** is active. Was that just a mistake when archiving?

There is no mistake. The relevant changes were in 4854a. 4517a is an abandoned branch I forgot to delete.

#38 - 02/08/2024 09:06 AM - Galya B

Although for a regression not related to #4517, there might have been a better place to do it and not this closed task. Why not open a branch under the task for r11266?

#39 - 02/08/2024 06:21 PM - Roger Borrello

Galya B wrote:

> > All I know is what I see...
>
> Since you're seeing, then we're on the same opinion that there is no regression in r14823. Go ahead and fix the r11266 regression.

I still believe there is a regression. The **env** will be passed in without any .ini on it, so you are always adding the extension for the check in the envIniMap. In the map is CounterAct=com.goldencode.p2j.util.StanzaIni@423c5404 so it doesn't ever match.

The envIniMap is created in the load method:

```
...
      boolean loaded = envAcc.load();

      if (loaded)
      {
         envs.put(env, envAcc);
      }
```

At this point, envs is the envIniMap, and the env in this load has not been changed to have an **.ini** extenstion in this method. This explains why the issue I noted related to r11266 has no bearing on the issue of the env not being found.

#40 - 02/08/2024 06:43 PM - Roger Borrello

- % Done changed from 100 to 90

Greg Shah wrote:

> Shouldn't we be testing to see if env already ends in .ini and removing it if it is there before assigning it to envName?

No, I don't think so. The original code, before Constantin made the current change was:

```
   public StanzaIni(String env, String directory, boolean flagNew, boolean readOnly)
   {
      this();

      name = env.endsWith(".ini") ? env : env + ".ini";
      if (directory != null)
      {
         workingDirectory = directory;
      }
      isNew = flagNew;

      isReadOnly = readOnly;

      // environment name should not contain .ini file extension
      envName = env;
   }
```

So it was intended that name should end in .ini and envName would not. I can change it to the below, but the results would be the same:

```
   public StanzaIni(String env, String directory, boolean flagNew, boolean readOnly)
   {
      contList = new LinkedList<Content>();
      sectMap = new TreeMap<String, Content>(String.CASE_INSENSITIVE_ORDER);

      // get driver type and read only property.
      ThinClient tc = ThinClient.getInstance();
      isChui = tc.isChui();

      name = env.endsWith(".ini") ? env : env + ".ini";
      if (directory != null)
      {
         workingDirectory = directory;
      }
      isNew = flagNew;

      isReadOnly = readOnly;

      // environment name should not contain .ini file extension
      envName = env;
   }
```

#41 - 02/08/2024 06:47 PM - Roger Borrello

- Status changed from Closed to Internal Test

- Assignee changed from Galya B to Roger Borrello

- % Done changed from 90 to 100

Created branch 4517b (dead-ended 4517a) and is at revision 14979. Not sure how to put this into review.

#42 - 02/08/2024 07:42 PM - Greg Shah

There is this comment // environment name should not contain .ini file extension and this code name = env.endsWith(".ini") ? env : env + ".ini"; which implies that sometimes env does have .ini in it. In that case, it seems like we should remove it **before** assignment in envName = env;.

#43 - 02/08/2024 11:48 PM - Roger Borrello

How about this? It's pretty explicit:

```
   public StanzaIni(String env, String directory, boolean flagNew, boolean readOnly)
   {
      contList = new LinkedList<Content>();
      sectMap = new TreeMap<String, Content>(String.CASE_INSENSITIVE_ORDER);

      // get driver type and read only property.
      ThinClient tc = ThinClient.getInstance();
      isChui = tc.isChui();

      if (env.endsWith(".ini"))
      {
         // environment name should not contain .ini file extension
         envName = env.substring(0, env.length() - 4);
         name = env;
      }
      else
      {
         envName = env;
         name = env + ".ini";
      }

      if (directory != null)
      {
         workingDirectory = directory;
      }

      isNew = flagNew;
      isReadOnly = readOnly;
   }
```

#44 - 02/09/2024 07:29 AM - Greg Shah

Yes, that seems reasonable.

Does your branch have the fix or fixes for the regression(s)?

#45 - 02/09/2024 08:55 AM - Roger Borrello

- Status changed from Internal Test to Review

Greg Shah wrote:

> Yes, that seems reasonable.
>
> Does your branch have the fix or fixes for the regression(s)?

I have both the fix for the regression (EnvironmentDaemon.java tested in the customer application) and the fix from way back in StanzaIni.java both in the branch.

I need to find appropriate testcases for regression testing in the new testcase model (which will have a bit of a learning curve for me).

#46 - 02/09/2024 08:56 AM - Greg Shah

Galya: Please review.

I will also review it.

#47 - 02/09/2024 09:58 AM - Greg Shah

Code Review Task Branch 4517b revision 14979 and 14980

I'm OK with the changes.

#48 - 02/13/2024 03:24 AM - Galya B

envIniMap is used in two places in EnvironmentDaemon: when using and unloading the environment.

Roger, you've introduced a regression, because the env is stored without .ini, but unload looks for it with the file ext.

Also the env can end with .ini and this is a perfectly valid use of LOAD:

```
LOAD "env.ini" DIR "/some-dir/" NEW BASE-KEY "INI".
USE "env.ini".
UNLOAD "env.ini"
```

If you remove the extension in StanzaIni EnvironmentDaemon.use will return a wrong value and also not find it in envIniMap.

Also, don't forget that the actual file always ends with .ini, but it's not added, when the env already has it in the name.

Please do more in-dept testing of all cases and share your testing methods.

#49 - 02/13/2024 09:33 AM - Roger Borrello

Thank you for the review. I'll take a look again.

#50 - 02/21/2024 04:09 PM - Roger Borrello

I have been absent from this task to get the testcases_v2 standardized. In doing so, I failed to find any testcases that might have covered LOAD/USE/PUT-KEY-VALUE/GET-KEY-VALUE. Galya, what testing were you able to do for your enhancements?

My assumption that the environment stored in the map does not contain the **.ini** extension is based upon the comments within the code. The Progress documentation makes reference to the search logic assuming that environment has the format *path\rootname.extension* (where *path* and *extension* are optional). Then the search logic makes it to the initialization file with "Else search for the initialization file *path\rootname.extension*. If found, load it." My interpretation of that is that the environment name should be devoid of the .ini extension when dealing with a stanza INI. So USE and UNLOAD should make sure to remove any extension when dealing with the envIniMap in the case of stanza INI.

Am I off base?

#51 - 02/21/2024 08:24 PM - Roger Borrello

Revision 15002 contains updates to make the mapping key standard (no .ini extension). The testcase I used was:

```
LOAD "env.ini" DIR "./data/" NEW BASE-KEY "INI".
USE "env.ini".
UNLOAD "env.ini".

LOAD "Env.ini" DIR "./data/" BASE-KEY "INI".
USE "ENV.ini".
UNLOAD "env.ini".

LOAD "env" DIR "./data/" BASE-KEY "INI".
USE "ENV.ini".
UNLOAD "env.ini".
```

However, there is the issue of case-sensitivity to be addressed. I would assume that each of those 3 sets should result in proper handling of env.ini, regardless of the case. Or should this take into account the case-sensitivity of the environment?

#52 - 02/22/2024 01:17 AM - Galya B

Roger Borrello wrote:

> Am I off base?

Yes.

Greg Shah wrote:

> The 4GL features LOAD, UNLOAD, USE, GET-KEY-VALUE, PUT-KEY-VALUE provide a mechanism to read and write key/value data from/into either INI files (any platform) or the Registry (Windows only). We support this fully.

This task is to port GET-KEY-VALUE, PUT-KEY-VALUE on non-Windows clients to ClientStorage. I've done a few fixes to the original registry logic, but it wasn't my focus. That's why I told you to reopen the original task and get yourself familiar with the state, since you're interested.

> My assumption that the environment stored in the map does not contain the .ini extension is based upon the comments within the code.

The code does a check for .ini in the name of the env, so obviously the comment you're referring to is not in sync. I haven't done any changes to StanzaIni, so I can't help you with investigating how it came to that.

I'm just trying to help you not introduce bugs by giving you a few hints about things I've seen and tested. Since you're committed to do changes here, just do your best to test them and ask the test team if you need support.

#53 - 02/22/2024 09:53 PM - Roger Borrello

Maybe I'm off base, but I am just trying to be helpful. There was a change introduced in the area of INI handling when #8074 was merged, which caused a significant regression in my customer's application, so I'm more than a little "interested." No INIs would load. I found that with this one change, I can prevent the failures in the customer's application:

```
=== modified file 'src/com/goldencode/p2j/util/EnvironmentDaemon.java'
--- old/src/com/goldencode/p2j/util/EnvironmentDaemon.java    2023-12-05 09:51:00 +0000
+++ new/src/com/goldencode/p2j/util/EnvironmentDaemon.java    2024-02-23 02:20:01 +0000
@@ -233,7 +233,8 @@

       if (loaded)
       {
+         envs.put(envAcc.getEnvironmentName(), envAcc);
-         envs.put(env, envAcc);
       }

       return loaded;
```

Perhaps I should have looked at #8074 more closely and posted there, to be clearer. After looking that task over, it looks like there wasn't a lot of attention paid to this update:

Galya B wrote:

> - EnvironmentDaemon: Fix for some inconsistencies in case-sensitivity. Handling several specific cases of PUT-KEY-VALUE deleting values / sections, as described here.

In any case, I've spent a lot of time tracking down this regression and taking it down a rabbit hole, since there wasn't any testing for INI in the testcases. When I introduced a testcase, there is other breakage since there are strange aspects to how Progress deals with LOAD/USE/UNLOAD in the areas of:

- The .ini extension doesn't seem to be required, it is added when needed
- The case of the INI value must match between LOAD/USE/UNLOAD.

We can carry this on in whichever task you'd like, but I'd appreciate helpful comments, specifically, why was that change necessary to env instead of the environment name?

#54 - 02/23/2024 01:38 AM - Galya B

Roger, create a test plan that includes the original behavior described in #4517-48, as well as your customer case, fix the comment. Apply the changes to the branch and let me know if you need me to review the code.

#55 - 02/23/2024 06:14 PM - Roger Borrello

The testing setup is taking longer than the code update, but I believe my updates in revision 15010 of this branch now handle what I've been able to test much better.

The rules I've found via checks on Progress are:

- The fname and envname are 2 distinct entities in LOAD/USE/UNLOAD
- In LOAD, envname is the fname stripped and fname is forced to .ini extension, if none given.
  - fname is the primary key. Lookups are done matching the envname to the filename and the first match is returned.
- In USE/UNLOAD, if an extension is passed it must match the fname, or fail.
  - If an extension is not passed, first envname is checked for. If not found, then .ini appended and checked against the fname.
- All checks are case-sensitive

I am trying to form tests that work in the new methodology, but so far I haven't had luck. Mostly because of my lack of 4GL skills. The testcase should:

1. use various combinations of names passed to LOAD and attempt USE/UNLOAD with matching/non-matching combinations.
2. utilize multiple LOAD@s in various orders, and verify using @GET-KEY-VALUE to validate values.
3. determine if a new file is created properly by writing/reading
4. read groups of sections and keys by passing null where appropriate
5. be performed in GUI and ChUI

There is a bug in the reading of a file when a section is the last line:

```
[section1]
key1 = value3txt1
key2 = value3txt2
[section2]
```

I have to add a newline at the end of the file for **section2** to be found.

Here is my currently manual testcase:

```
def var path as char init "testcases_v2/abl/stanza_ini".
def var inidir as char.
inidir = "z:/" + path.
if opsys = "unix" then
   inidir= "/home/rfb/" + path.
if opsys <> "win32" then
   message opsys "is an unsupported OS".

DEFINE VARIABLE keyval AS CHARACTER NO-UNDO FORMAT "x(128)".

/* Normal checks */
load "env.1.inifile" dir inidir base-key "INI".
use "env.1.inifile".
GET-KEY-VALUE SECTION "section1" KEY "key1" VALUE keyval.
message "keyval="keyval.
use "".
unload "env.1.inifile".

load "env1.ini" dir inidir base-key "INI".
use "env1".
GET-KEY-VALUE SECTION "section1" KEY "key1" VALUE keyval.
message "keyval="keyval.
use "".
unload "env1".

load "env1" dir inidir base-key "INI".
use "env1.ini".
GET-KEY-VALUE SECTION "section1" KEY "key1" VALUE keyval.
message "keyval="keyval.
use "".
unload "env1".

load "env1.ini" dir inidir base-key "INI".
use "env1.ini".
GET-KEY-VALUE SECTION "section1" KEY "key1" VALUE keyval.
message "keyval="keyval.
use "".
unload "env1.ini".

load "env3.txt" dir inidir base-key "INI".
load "env3" dir inidir base-key "INI".
use "env3".
GET-KEY-VALUE SECTION "section1" KEY "key1" VALUE keyval.
message "keyval="keyval.

GET-KEY-VALUE SECTION "" KEY "key1" VALUE keyval.
message "keyval="keyval.

GET-KEY-VALUE SECTION "section1" KEY "" VALUE keyval.
message "keyval="keyval.

GET-KEY-VALUE SECTION "section" KEY DEFAULT VALUE keyval.
message "keyval="keyval.
unload "env3".

use "".
GET-KEY-VALUE SECTION "section1" KEY "key1" VALUE keyval.
message "keyval="keyval.
```

#56 - 02/26/2024 06:02 AM - Greg Shah

Roger: Please create a new task in the base language project for this work. Copy your #4517-55 content there and keep working. You should look at the existing unit tests to understand how to structure the code. Make sure you read Writing 4GL Testcases, especially the section on "Automated/Batch Execution".

#57 - 02/27/2024 11:33 AM - Roger Borrello

- Related to Bug #8353: Fully support stanza INI LOAD/USE/UNLOAD scenarios added

#58 - 03/28/2024 01:47 PM - Greg Shah

- Status changed from Review to Closed

Files

preferences.png	77.4 KB	06/16/2023	Galya B