Base Language - Feature #3292

i18n improvements

04/27/2017 03:05 PM - Greg Shah

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

Description

Related issues:

Related to Base Language - Feature #3753: I18N additions	**Closed**

History

#1 - 04/27/2017 03:06 PM - Greg Shah

See #3257-15:

SESSION:CPSTREAM
NO-MAP stream output option
BINARY (in OUTPUT TO)

#2 - 06/09/2017 03:43 PM - Ovidiu Maxiniuc

- Status changed from New to WIP

Apparently work for first two can be delayed.
I started investigations on BINARY. I was expecting the output to be really binary but, from my tests, the only difference I could note is the EOLN encoding on Windows: if BINARY is specified, Windows client output the file as a Linux client.

Documentation is really vague: *output to be written directly without any conversion or interpretation*. I don't know what else *conversion* and *interpretation* could mean. Used in INPUT FROM, the documentation states that *NUL (\0) terminates character strings, and other control characters are interpreted as expected for the operating system*. I found no evidence of the NUL character to be written at the end of strings in either output modes.

#3 - 06/12/2017 04:08 PM - Ovidiu Maxiniuc

I propagated the binary attribute from RemoteStream to actual FileStream that does the string encodings. The preliminary tests work fine.

One question: I don't know exactly whether FileStream is a client side only or it can be used on server side too. I need this in order to correctly detect the OS we are emulating - so I either need access directly to registry or use the ThinClient instance.

#4 - 06/12/2017 04:26 PM - Greg Shah

Instances of FileStream are only ever constructed and used on the client side.

#5 - 06/22/2017 03:05 PM - Ovidiu Maxiniuc

- Assignee set to Ovidiu Maxiniuc

- Start date deleted (04/27/2017)

I created last week 3292a branch for this task. I committed today after comparative tests between Windows and Linux. Current revision is 11153.

#6 - 07/28/2017 10:57 AM - Greg Shah

Code Review Task branch 3292a Revision

I did check in some minor changes for code formatting.

1. My only concern with the basic approach is just that it adds an extra trip to the client side when running on Windows and a CR is encountered. I guess this is not that much overhead in the common case of a file where the text to CR ratio is high. We can probably leave this as is.

2. The import com.goldencode.p2j.ui.chui.*; in FileStream is not a dependency we want to have. We have two ways to handle this:

- The result returned by TC.isUnderWindows() comes from EnvironmentOps.isUnderWindowsFamily() (either directly when TC is running on the server, or indirectly through an up-call to ServerPropertiesInspector.isUnderWindowsFamily()). The EnvironmentOps.isUnderWindowsFamily() will first try to lookup if we have an OS name override in the directory. If that doesn't exist, it will downcall to the client to lookup the os.name Java system property.
- The result from PlatformHelper.isUnderWindowsFamily() simply bases its result on the os.name Java system property of the local system (in this case the client).

The up-call to ServerPropertiesInspector.isUnderWindowsFamily() would be used if we need to allow the customer to configure our output to match the original 4GL system instead of the current client runtime system (the original system was Windows but they are running FWD on Linux). This is useful if the output would be incompatible or broken if we don't honor this.

The PlatformHelper.isUnderWindowsFamily() would be used if the original system doesn't matter and we just always need to operate with the current platform in mind.

#7 - 07/28/2017 11:06 AM - Greg Shah

> output to be written directly without any conversion or interpretation. I don't know what else conversion and interpretation could mean

I believe this may reference the various ways that codepage conversion may occur.

If you set SESSION:CPSTREAM to a value that is different from SESSION:CPINTERNAL, then there is supposed to be some amount of implicit/automatic codepage conversion that occurs.

One can also specify the CONVERT [SOURCE codepage] [TARGET codepage] which can manually override the implicit behavior.

I think it is important to test both these cases and implement any modification of the implicit/explicit codepage conversion that actually occurs based on BINARY being set. Today FWD does not implement much in this area. But we do have the basic implementation in a skeleton form. We do track if conversion is needed (e.g. NO-CONVERT option), but we don't modify this decision based on the BINARY flag.

Considering this is checked in multiple places, it would be important to know if BINARY has the same affect everywhere.

#8 - 08/15/2017 10:11 AM - Greg Shah

From what I see, branch 3292a has been rebased from trunk 11157, but otherwise is not changed from my previous code review. Correct?

#9 - 08/15/2017 10:25 AM - Ovidiu Maxiniuc

Greg Shah wrote:

> From what I see, branch 3292a has been rebased from trunk 11157, but otherwise is not changed from my previous code review. Correct?

Right. I did not apply change(s) from note 6 yet.

#10 - 08/15/2017 04:29 PM - Greg Shah

> > From what I see, branch 3292a has been rebased from trunk 11157, but otherwise is not changed from my previous code review. Correct?
>
> Right. I did not apply change(s) from note 6 yet.

OK, no problem. The runtime changes can be made later, in another branch.

#11 - 10/27/2017 04:09 PM - Ovidiu Maxiniuc

I have chosen the EnvironmentOps.isUnderWindowsFamily() instead of TC.isUnderWindows(). The returned value can be configured in registry and defaulting to host OS JVM runs on.

Regarding the connection between BINARY and CP conversion. I did a few tests (well, there are a lot of code pages in convmap.cp and I cannot test all combinations) but I could not find any difference in converted and binary output except for the extra 0x0D character that doubles 0x0A when in non binary. I guess the ABL Reference is not really exact for this option, too.

I created a new branch (3292b) for this work but since the changes are limited to 3 LoC in FileStream, I will merge it into 1830b.

#12 - 11/01/2017 02:30 PM - Greg Shah

- Status changed from WIP to Closed

- % Done changed from 0 to 100

The runtime implementation for this task was written in branch 1830b and was merged to trunk as revision 11188.

BINARY support is considered "full". The other features in this task turned out to be unneeded and were not implemented.

#13 - 10/20/2018 09:11 AM - Greg Shah

- Related to Feature #3753: I18N additions added