Base Language - Feature #6431

implement support to allow CPINTERNAL set to UTF-8 and CP936

05/26/2022 07:43 PM - Greg Shah

Status:	WIP	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Theodoros Theodorou	**% Done:**	50%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to Base Language - Feature #4761: I18N phase 3	**New**
Related to User Interface - Bug #7657: 8-bit character entry problem in ChUI	**New**

History

#1 - 06/08/2022 09:36 PM - Greg Shah

- Subject changed from implement support to allow CPINTERNAL set to UTF-8 to implement support to allow CPINTERNAL set to UTF-8 and CP936

#2 - 04/05/2023 12:51 PM - Greg Shah

- Assignee set to Joe Davis

We have a customer that needs to be able to set UTF-8 or CP963 as the CPINTERNAL. We may have some problems with doing this today with UTF-8 for which we have some really old nasty workarounds that sometimes treat UTF-8 as 8859-1 (e.g. see I18nOps.getCharsetNative()).

#3 - 07/05/2023 02:51 PM - Greg Shah

- Assignee changed from Joe Davis to Theodoros Theodorou

#4 - 08/02/2023 05:36 PM - Theodoros Theodorou

I would like to ask for some guidance for this task. As I understand, we are trying to emulate a keyboard based on the -cpinternal? How should we handle UTF-8 and CP936? How can I test the desired functionality?

#5 - 08/02/2023 05:41 PM - Theodoros Theodorou

- Status changed from New to WIP

#7 - 08/03/2023 10:10 AM - Greg Shah

> As I understand, we are trying to emulate a keyboard based on the -cpinternal?

Not exactly. In the 4GL, the CPINTERNAL defines the code page used as the default inside the 4GL process. This can affect many things, but generally it affects any kind of encoding, decoding and comparison of character/text data. This means that the same data in a character variable will be interpreted differently based on setting CPINTERNAL differently at the 4GL session startup (via command line option). You can only set it at startup, that attribute is not writable.

Since it is the default for the 4GL session, this means that any of the other CP* attributes that are not explicitly set on 4GL session startup will automatically be assumed to be the same as whatever CPINTERNAL is set to. The UI processing of the terminal is defined by CPTERM and read/write of streams is set by CPSTREAM, so if these are set explicitly they have an affect on how keyboard processing or terminal I/O is encoded.

There is a useful discussion in #4761. Please read that, review the linked tasks and the source code for our existing support.

> How should we handle UTF-8 and CP936?

Understand our current support and then we can discuss this based on your proposals or questions.

> How can I test the desired functionality?

The same way we would for any other feature.

We write tests to check how this encoding/decoding/collation affects different aspects of the 4GL. Many of these test already exist. You may need to help write more. Once we have sufficient tests, you run them in the 4GL using different CP values and then run them in FWD with the same CP values until our support is complete and correct.

#8 - 08/03/2023 10:10 AM - Greg Shah

- Related to Feature #4761: I18N phase 3 added

#9 - 08/17/2023 07:32 AM - Theodoros Theodorou

Can you please help me by giving me a 4GL code to do I/O in different charsets? (UTF-8 / CP936)

#10 - 08/17/2023 07:36 AM - Greg Shah

Look at statements like PUT, EXPORT, IMPORT, INPUT FROM, OUTPUT TO. I/O in different character sets is the same as I/O, but just with a different setting for CPINTERNAL and/or CPSTREAM. Just write code to read/write from these sources and then setup tests to check that the character set processing is working.

#11 - 08/17/2023 04:49 PM - Theodoros Theodorou

```
OUTPUT TO "out.txt".

DEFINE VARIABLE i AS integer NO-UNDO.

DO i = 0 TO 65535:
    message CHR(i, "ISO8859-1").
END.

OUTPUT CLOSE.
```

Is the above correct for printing all the characters in a charset? As I have read, a UTF-8 character can use up to 4 bytes, ISO8859-1 and CP936 can use up to two bytes. ISO8859-1 and UTF-8 print some results but CP936 returns the following error in OE:

```
Code page conversion table for ISO8859-1 to CP936 was not found in convmap.cp. (6063)
CHR could not find the appropriate conversion table in convmap.cp. (1586)
```

If I try to set prowin -cpinternal CP936 -p example.p I get the error:

```
Code page conversion table for ISO8859-1 to CP936 was not found in convmap.cp. (6063)
```

#12 - 08/18/2023 05:38 PM - Theodoros Theodorou

I think this task will drive me crazy.

```
message chr(181, "ISO8859-1") = 'µ'.    // true
message chr(181, "UTF-8") = 'Âµ'.       // true
message chr(181, "UTF-16") = 'µ'.       // true
message chr(181, "UTF-32") = 'µ'.       // true
```

Do you guys have any clue why OE adds an Â for UTF-8?

```
        message(isEqual(chr(new integer(181), new character("ISO8859-1")), new character("Âµ")));
        message(isEqual(chr(new integer(181), new character("UTF-8")), new character("Ã‚Âµ")));
        message(isEqual(chr(new integer(181), new character("UTF-16")), new character("Âµ")));
        message(isEqual(chr(new integer(181), new character("UTF-32")), new character("Âµ")));
```

As I can see in FWD we replace µ with Âµ. Do you guys have any clue on what is happening here? Where is this replacement happening in our conversion?

#13 - 10/05/2023 06:27 PM - Theodoros Theodorou

I need help for understanding what needs to be done. Codepages look very tricky. Progress OE returns unexpected results. For example, I tried to use the chr() function which prints all the 255 characters correctly with ISO8859-1 but when I use -cpinternal UTF-8, it prints only the first 128 characters (see testcases/cp/cpinternal/chr.p).

I tried reading greek (unicode) characters from a file (INPUT FROM) and writing them to a file (OUTPUT TO) and it works with ISO8859-1 if the output file is opened as UTF-8.

I tried printing some chinese characters using message() and -cprcodein UTF-8 -cpinternal UTF-8 and it worked. I tried printing greek characters and it didn't work... I tried using a mix of chinese and greek characters and it worked. If I write these in a file it always works.

I don't know what should I test and how should I do it correctly. I think the best way to proceed is to solve real customer project issues instead of spending hours trying to find randomly something meaningful to test and implement.

#14 - 10/05/2023 06:44 PM - Ovidiu Maxiniuc

Indeed, this is tricky. I think this is the first time I see -cprcodein OE parameter. If I understand this parameter correctly, it has no correspondent in FWD. Internally, all our code use Unicode. Also, this refers to text 'embedded' (constant literals) in compiled code, not read from a file or keyboard.

IIRC, OE will display correctly only the valid characters in current -cpinternal CP. Forcing to print other characters will fail, causing incorrect characters to be displayed on screen, if any.

#15 - 10/06/2023 09:05 AM - Greg Shah

Are all your notes from #6431-13 based on OE testing? I don't want to mix any FWD results together at this point, it it too confusing.

> I don't know what should I test and how should I do it correctly.

Let's work together to figure this out. Perhaps we should have a call to discuss it.

> I think the best way to proceed is to solve real customer project issues instead of spending hours trying to find randomly something meaningful to test and implement.

We've been doing this already and it has resulted in the mess we currently have. This task and the relate I18N tasks are about developing a full understanding of the features and implementing them properly. I understand it is hard but we need to do this. The "fix bugs as we go" model is a plan for never finishing the work. We have real deadlines and need to get this done.

#16 - 10/06/2023 12:03 PM - Theodoros Theodorou

Ovidiu Maxiniuc wrote:

> Indeed, this is tricky. I think this is the first time I see -cprcodein OE parameter. If I understand this parameter correctly, it has no correspondent in FWD. Internally, all our code use Unicode. Also, this refers to text 'embedded' (constant literals) in compiled code, not read from a file or keyboard.

I tried to use something like message '□□ αβγ' which has unicode characters in it. The source file is also saved in UTF-8. As far as I understand, using -cprcodein UTF-8 tells OE that the source file should be read as UTF-8.

#17 - 10/06/2023 12:10 PM - Theodoros Theodorou

Greg Shah wrote:

> Are all your notes from #6431-13 based on OE testing? I don't want to mix any FWD results together at this point, it it too confusing.

Yes, I focused on OE testing too because this alone is very tricky.

> I don't know what should I test and how should I do it correctly.

> Let's work together to figure this out. Perhaps we should have a call to discuss it.

I think this is a great idea. We can schedule a call early next week if you like.

> I think the best way to proceed is to solve real customer project issues instead of spending hours trying to find randomly something meaningful to test and implement.

> We've been doing this already and it has resulted in the mess we currently have. This task and the relate I18N tasks are about developing a full understanding of the features and implementing them properly. I understand it is hard but we need to do this. The "fix bugs as we go" model is a plan for never finishing the work. We have real deadlines and need to get this done.

I understand. I need help at least to find out what needs to be done and create meaningful tests for OE (I do not have strong experience of the OE 4gl yet). If I have the tests for OE, I think I will be able to handle the java FWD part.

#18 - 10/06/2023 12:17 PM - Greg Shah

> As far as I understand, using -cprcodein UTF-8 tells OE that the source file should be read as UTF-8.

No, I don't think that is correct.

"r-code" is the tokenized compiler output. When you run the COMPILE language statement, you turn a source program (often a .p file) into an r-code program (.r). r-code is a binary format, similar to Java byte code. The source file encoding is not related to the encoding of the r-code file. The COMPILE statement determines the encoding of the r-code file, which can be set from options in the language statement or by default from the cprcodeout setting of the 4GL session in which the COMPILE was executed.

Setting cprcodein changes the default used when reading r-code back in, which has to be done in order to execute those programs. In FWD, we have no equivalent since the byte code of a class file is not dependent upon a separate encoding setting to be loaded. All the strings are stored as UTF-8 in .class files.

#19 - 10/18/2023 04:09 PM - Theodoros Theodorou

I am trying to build a prorgam in OE that generates tests. I generated a file located at testcases/cp/cpinternal/iso8859-1/utf-8_to_iso8859-1.p as an example of what is generated. It generates something like this:

assert-true(asc(chr(21728, 'ISO8859-1', 'UTF-8')) eq 84, 'chr with id = 21728 should be 84', {&line-number}).

It converts the character with id 21728 from UTF-8 to ISO8859-1 which gives the id 84. The program that I run in OE automatically generates that id 21728 should be 84 after the conversion. My goal is to test that FWD behaves the same as OE. The above code returns different results when it is executed with e.g. "-cpinternal UTF-8". For example, there are ids which will produce an empty string or "?" with UTF-8.

What do you think about my idea for testing asc() and chr() with different -cpinternals? Should I continue or should I look for something else?

#20 - 10/18/2023 04:17 PM - Greg Shah

You could make this data driven. The idea is to capture the inputs and results in an array even in a data file. Then the core of the program could be pretty small code that just iterates through the data and checks each case.

If need you can still write code to generate the data that is the inputs and results. My point is that these should not be hard coded in separate lines but would be more maintainable if it was just stored as data.

#21 - 10/18/2023 04:19 PM - Greg Shah

> What do you think about my idea for testing asc() and chr() with different -cpinternals?

I think we must do that to get all of this correct. Perhaps the array of results needs to encode results for a list of different -cpinternal values and we can compare the correct output in FWD depending on our -cpinternal value.

#22 - 10/18/2023 04:56 PM - Theodoros Theodorou

Greg Shah wrote:

> You could make this data driven. The idea is to capture the inputs and results in an array even in a data file. Then the core of the program could be pretty small code that just iterates through the data and checks each case.
>
> If need you can still write code to generate the data that is the inputs and results. My point is that these should not be hard coded in separate lines but would be more maintainable if it was just stored as data.

Yes, that is why I thought of using something like a Java HashMap in my "test generator". I want to be able to know which UTF-8 characters are converted to each ISO8859-1 character and put them in an array or list. Is there something similar to the HashMap in 4gl other than OpenEdge.Core.Collections?

#23 - 10/18/2023 05:02 PM - Theodoros Theodorou

Greg Shah wrote:

> > What do you think about my idea for testing asc() and chr() with different -cpinternals?
>
> I think we must do that to get all of this correct. Perhaps the array of results needs to encode results for a list of different -cpinternal values and we can compare the correct output in FWD depending on our -cpinternal value.

I was thinking about creating separate test files for each -cpinternal to make the tests smaller and more readable. I thought about adding something like the following in the header of the file:

```
if session:cpinternal ne 'ISO8859-1' then do:
   message '-cpinternal is not ISO8859-1'.
   quit.
end.
```

What do you think?

#24 - 10/18/2023 05:11 PM - Greg Shah

We will be running these as unit tests using our ABLUnit support. At some point, your testcases will be reworked to fit into that model. So we would not want to do things like message or quit.

The general concept that we have results that are specific to a -cpinternal is certainly valid. But I think we would be better off with a common test that knows the different results it should get based on the current -cpinternal. Doing it that way makes it much easier to run these as unit tests since we always run the same test set. We just run it multiple times, with different -cpinternal values but the same tests are run.

If we do it the way you propose, then we would have to know which tests to run depending on the -cpinternal value. That seems unnecessary when we can just make the test code more flexible. If you want to load a different set of results based on the -cpinternal, that is a perfectly good choice. I prefer if that data is coded into the 4GL source, rather than as a data file. Data file access at runtime in FWD means that we would have to place it in the right location, which is more work and more fragile than just having the right data built into the 4GL code.

#25 - 10/18/2023 05:18 PM - Greg Shah

Theodoros Theodorou wrote:

> Greg Shah wrote:
>
> > You could make this data driven. The idea is to capture the inputs and results in an array even in a data file. Then the core of the program could be pretty small code that just iterates through the data and checks each case.
> >
> > If need you can still write code to generate the data that is the inputs and results. My point is that these should not be hard coded in separate lines but would be more maintainable if it was just stored as data.
>
> Yes, that is why I thought of using something like a Java HashMap in my "test generator". I want to be able to know which UTF-8 characters are converted to each ISO8859-1 character and put them in an array or list. Is there something similar to the HashMap in 4gl other than OpenEdge.Core.Collections?

There are 3 common approaches:

1. Use a temp-table. This is the most common way a 4GL dev would do it. It allows a flexible record to be defined and then you have a "list" of them as your table. You can use their query capability to traverse it.
2. Use an array of OO 4GL object instances. In this case, the OO class you create is just a simple set of public data members. You don't have to have any methods. Then just create an array of these instances (def var my-object-array as class some.oo.namespace.TestAndResultData extent.) and fill it up with the expected input/results.
3. Use matched sets of arrays. This is only useful for simple data sets but if you only have 3 values to track (input val 1, input val 2 and result), then you can create 3 independent arrays but make them the same length. Then the same element index in each array is related. This is not a clean way to do it but before OO 4GL, it was the way you would do it if you really wanted to avoid temp-table usage.

#26 - 10/18/2023 05:27 PM - Ovidiu Maxiniuc

Theodoros Theodorou wrote:

> I am trying to build a prorgam in OE that generates tests. I generated a file located at testcases/cp/cpinternal/iso8859-1/utf-8_to_iso8859-1.p as an example of what is generated. It generates something like this:
>
> assert-true(asc(chr(21728, 'ISO8859-1', 'UTF-8')) eq 84, 'chr with id = 21728 should be 84', {&line-number}).
>
> It converts the character with id 21728 from UTF-8 to ISO8859-1 which gives the id 84. The program that I run in OE automatically generates that id 21728 should be 84 after the conversion.

This example is not really representative. There is some error during conversion. The T character has code 84 in both ISO8859-1 and UTF-8 CPs. You can test this by just replacing 84 in the code above. See what that character actually is here: https://www.unicodepedia.com/unicode/cjk-unified-ideographs-extension-b/21728/cjk-unified-ideograph-21728/.

> My goal is to test that FWD behaves the same as OE. The above code returns different results when it is executed with e.g. "-cpinternal UTF-8". For example, there are ids which will produce an empty string or "?" with UTF-8.

ISO8859-1 can represent only 256 characters. For those ids whose character cannot be represented the unknown character is returned.

> What do you think about my idea for testing asc() and chr() with different -cpinternals? Should I continue or should I look for something else?

If you use chr function with 3 parameters (expression, targetCP, sourceCP) I do not think the cpinternal plays any role (it is overwritten by last parameter). See https://docs.progress.com/bundle/abl-reference/page/CHR-function.html.

#27 - 10/18/2023 05:48 PM - Theodoros Theodorou

Ovidiu Maxiniuc wrote:

> Theodoros Theodorou wrote:
>
> > I am trying to build a prorgam in OE that generates tests. I generated a file located at testcases/cp/cpinternal/iso8859-1/utf-8_to_iso8859-1.p as an example of what is generated. It generates something like this:

> assert-true(asc(chr(21728, 'ISO8859-1', 'UTF-8')) eq 84, 'chr with id = 21728 should be 84', {&line-number}).
>
> It converts the character with id 21728 from UTF-8 to ISO8859-1 which gives the id 84. The program that I run in OE automatically generates that id 21728 should be 84 after the conversion.

This example is not really representative. There is some error during conversion. The T character has code 84 in both ISO8859-1 and UTF-8 CPs. You can test this by just replacing 84 in the code above. See what that character actually is here: https://www.unicodepedia.com/unicode/cjk-unified-ideographs-extension-b/21728/cjk-unified-ideograph-21728/.

I know this doesn't make sense but this is what OE returns. If you run message asc(chr(21728, 'ISO8859-1', 'UTF-8')). it will print 84.

> What do you think about my idea for testing asc() and chr() with different -cpinternals? Should I continue or should I look for something else?

If you use chr function with 3 parameters (expression, targetCP, sourceCP) I do not think the cpinternal plays any role (it is overwritten by last parameter). See https://docs.progress.com/bundle/abl-reference/page/CHR-function.html.

Again, I know this doesn't make any sense but I observed different results with different code pages. e.g. with UTF-8 as cpinternal, the characters that were not "" or "?" were fewer compared to ISO8859-1 (default).

#28 - 10/18/2023 05:52 PM - Theodoros Theodorou

Greg Shah wrote:

> We will be running these as unit tests using our ABLUnit support. At some point, your testcases will be reworked to fit into that model. So we would not want to do things like message or quit.
>
> The general concept that we have results that are specific to a -cpinternal is certainly valid. But I think we would be better off with a common test that knows the different results it should get based on the current -cpinternal. Doing it that way makes it much easier to run these as unit tests since we always run the same test set. We just run it multiple times, with different -cpinternal values but the same tests are run.
>
> If we do it the way you propose, then we would have to know which tests to run depending on the -cpinternal value. That seems unnecessary when we can just make the test code more flexible. If you want to load a different set of results based on the -cpinternal, that is a perfectly good

> choice. I prefer if that data is coded into the 4GL source, rather than as a data file. Data file access at runtime in FWD means that we would have to place it in the right location, which is more work and more fragile than just having the right data built into the 4GL code.

OK, noted. Is there anything else I should have in mind when developing tests regarding ABLUnit support?

#29 - 10/27/2023 04:37 PM - Theodoros Theodorou

- File out.rar added

Greg Shah wrote:
I prefer if that data is coded into the 4GL source, rather than as a data file. Data file access at runtime in FWD means that we would have to place it in the right location, which is more work and more fragile than just having the right data built into the 4GL code.

Hello Greg, I tried to add the data into the 4gl source but I get the error Acode Segment has exceeded its limit of 4194304 bytes, in at line #-19945. (3307) in OE. You can try running out.p which is attached. Don't try to open this file using the Progress Developer Studio because it will crash. The test testcases/cp/cpinternal/iso8859-1/utf-8_to_iso8859-1.p didn't have issues running. How should I proceed?

#30 - 10/30/2023 06:51 AM - Marian Edu

Theodoros Theodorou wrote:

> Greg Shah wrote:
> I prefer if that data is coded into the 4GL source, rather than as a data file.

I would rather disagree with this one, especially when we talk about milion code points per codepage (UTF) and have to test all combinations of supported code pages - even if we narrow down the list of supported ones.

Keeping only numbers (code points) in those data files, as opposed to the actual 'character', looks to me like a valid approach - we already use external data files for other test cases and it was not an issue finding them until now :)

Just don't use absolute values, relative values together with search are known to work just fine.

#31 - 10/30/2023 06:54 AM - Marian Edu

BTW, Daniel is working on something more or less the same for the I18N part of the base language unit tests - not specific to those particular code pages but the approach is similar I guess, use CHR and ASC to convert back and forth from one code point to another in different code pages. If this part is to be done under this task then we can probably simply stop working on it on our end, or we just do it for validation - like a second opinion :)

#32 - 10/30/2023 07:30 AM - Greg Shah

If there is a reasonably small data set, then keeping it in the source code should be the standard approach. I agree that large data sets should be externalized. Marian's idea is the standard there.

Marian: I prefer if Daniel writes the tests. I did not know he was working on them yet, so Theodoros has been trying to get this figured out. Theodoros is new to 4GL development so let's coordinate the work such that Daniel prioritizes I18N test writing that can enable Theodoros to develop a specification of what has to be written to implement compatible support in FWD. We especially need to know the timing. Can Theodoros run these tests today/soon?

Theodoros: Please document your findings so far. Help to ensure that the tests written by Daniel will meet the needs of the I18N tasks including this one. Execute the tests and write up the specification here in Redmine.

#33 - 10/30/2023 07:25 PM - Theodoros Theodorou

Greg Shah wrote:

> Theodoros: Please document your findings so far. Help to ensure that the tests written by Daniel will meet the needs of the I18N tasks including this one. Execute the tests and write up the specification here in Redmine.

I have created a test at testcases/cp/cpinternal/chr_and_asc.p. The differences between the different cpinternal values can be seen by using the linux command diff testcases/cp/cpinternal/utf-8_to_iso8859-1_using_iso8859-1.txt testcases/cp/cpinternal/utf-8_t
o_iso8859-1_using_utf-8.txt. I was not able to run the tests using FWD yet due to some configuration errors. I will try to resolve the issues tomorrow and let you know what is supported by FWD and what is not.

I also wanted to ask how to run an application using -cpinternal CP936. I get the error message Code page conversion table for CP936 to ISO8859-1 was not found in convmap.cp.

#34 - 10/31/2023 06:59 AM - Greg Shah

> I also wanted to ask how to run an application using -cpinternal CP936. I get the error message Code page conversion table for CP936 to ISO8859-1 was not found in convmap.cp.

That probably just means that the CP936 hasn't been installed in that OE installation.

#35 - 10/31/2023 07:06 AM - Marian Edu

Greg Shah wrote:

> I also wanted to ask how to run an application using -cpinternal CP936. I get the error message Code page conversion table for CP936 to ISO8859-1 was not found in convmap.cp.

That probably just means that the CP936 hasn't been installed in that OE installation.

No, this means there is no conversion possible between `western europe` and `chineze` code pages :)

If it's a character mode client set the -cpterm and maybe also -cpstream to cp936 otherwise is using the installation defaults - in your case seems to be iso8859-1.

#36 - 10/31/2023 05:45 PM - Theodoros Theodorou

Marian Edu wrote:

> If it's a character mode client set the -cpterm and maybe also -cpstream to cp936 otherwise is using the installation defaults - in your case seems to be iso8859-1.

I think -rcodein is required too. When I run an example like prowin -p test.p -cpinternal CP936 -cpterm CP936 -cpstream CP936 -cprcodein CP936 to print something like message 'test', I see some errors in the GUI, Unable to open word-break table file 247. and The word-rule file specified by the -ttwrdrul parameter is invalid. (9258). Do you know how can I solve these too?

Greg, if all of these parameters should be tested with different combinations, this means that my tests are not complete and this makes testing even more complicated. These means that many -cp* parameters can change the outcome. How should I proceed? Should we consider that everything defaults to iso8859-1 or is this a false assumption?

#37 - 10/31/2023 08:26 PM - Greg Shah

I think the -cpterm and -cpstream options are purely to avoid an unexpected conversion when you output data to a stream. These just need to be consistent with the -cpinternal otherwise there is an implicit conversion to the default -cpterm or -cpstream (depending on the method used for output to the file).

#38 - 10/31/2023 08:41 PM - Greg Shah

From Theodoros:

The program which creates the data in files looks like this:

```
def var i as int no-undo.
def var a as int no-undo.
def var s as char no-undo.

output to value('utf-8_to_iso8859-1_using_' + lc(session:cpinternal) + '.txt').

do i = 0 to 0x10FFFF:
    a = asc(chr(i, 'ISO8859-1', 'UTF-8')).
    message string(i) + " " + string(a).
end.

output close.

output to value('iso8859-1_to_utf-8_using_' + lc(session:cpinternal) + '.txt').

do i = 0 to 0x10FFFF:
    a = asc(chr(i, 'UTF-8', 'ISO8859-1')).
    message string(i) + " " + string(a).
end.

output close.
```

#39 - 11/01/2023 03:56 AM - Marian Edu

Theodoros Theodorou wrote:

> I think -rcodein is required too.

Generally speaking all codepages used must be set so conversion to be possible - when needed - best is to really use the same codepage for all those so no conversion happens at runtime, the only accepted exception is for databases connected that can have different codepages (normally some version of UTF).

> When I run an example like prowin -p test.p -cpinternal CP936 -cpterm CP936 -cpstream CP936 -cprcodein CP936 to print something like message 'test', I see some errors in the GUI, Unable to open word-break table file 247. and The word-rule file specified by the -ttwrdrul parameter is invalid. (9258). Do you know how can I solve these too?

Since you're not using the -ttwrdrul parameter anyway I would say this is exactly the case from this KB entry - [[

https://community.progress.com/s/article/Word-rule-errors-2736-9258-starting-the-procedure-editor]].

#40 - 11/01/2023 04:06 AM - Marian Edu

Greg Shah wrote:

> From Theodoros:
>
> The program which creates the data in files looks like this:

Strings/characters are still just a succession of bytes, in order to interpret those correctly one needs to know the encoding/codepage used. I think we've been down that path before, now I've decided to simply use code points (numbers) in those data files to get the mapping used by ABL so we can validate that against FWD implementation.

In your example all strings variables are stored using the cpinternal codepage and then when written out to a file are converted (if needed) to the cpstream encoding so if you ran those with different start-up settings you'll get different results in those files. Aka, you must known what was the codepage you used to generate the data in files and then read them using the same encoding otherwise it's a nice Babel like experience :)

#41 - 11/01/2023 11:21 AM - Theodoros Theodorou

Marian Edu wrote:

> Since you're not using the -ttwrdrul parameter anyway I would say this is exactly the case from this KB entry - [[https://community.progress.com/s/article/Word-rule-errors-2736-9258-starting-the-procedure-editor]].

I tried removing the DLC system environmental variable but it didn't help.

#42 - 11/01/2023 03:17 PM - Theodoros Theodorou

Marian, I am trying to run something like asc(chr(i, 'CodePageA', 'CodePageB')). Which parameters should I change? Only -cpinternal -cpterm -cpstream or I need to change other parameters too?

Also, is there any other way to change the "system wide" default codepage instead of setting the parameters one by one?

#43 - 11/16/2023 06:03 PM - Theodoros Theodorou

Hello, I wrote a small test (chr_and_asc.p) which tests the conversion of a character from one code page to another (e.g. i = 0 to 0x10FFFF, asc(chr(i, 'ISO8859-1', 'UTF-8'))). As I noticed, there are thousands of entries which do not match the OE behavior (used cpinternal=utf-8):

```
UNEXPECTED_ERROR: Code page error for i = 480. Expected 1 but got -1 #19
UNEXPECTED_ERROR: Code page error for i = 481. Expected 1 but got -1 #19
```

```
UNEXPECTED_ERROR: Code page error for i = 482. Expected 1 but got -1 #19
UNEXPECTED_ERROR: Code page error for i = 483. Expected 1 but got -1 #19
UNEXPECTED_ERROR: Code page error for i = 484. Expected 1 but got -1 #19
UNEXPECTED_ERROR: Code page error for i = 485. Expected 1 but got -1 #19
UNEXPECTED_ERROR: Code page error for i = 486. Expected 1 but got -1 #19
UNEXPECTED_ERROR: Code page error for i = 487. Expected 1 but got -1 #19
UNEXPECTED_ERROR: Code page error for i = 488. Expected 1 but got -1 #19
UNEXPECTED_ERROR: Code page error for i = 489. Expected 1 but got -1 #19
UNEXPECTED_ERROR: Code page error for i = 490. Expected 1 but got -1 #19
UNEXPECTED_ERROR: Code page error for i = 491. Expected 1 but got -1 #19
UNEXPECTED_ERROR: Code page error for i = 492. Expected 1 but got -1 #19
UNEXPECTED_ERROR: Code page error for i = 493. Expected 1 but got -1 #19
UNEXPECTED_ERROR: Code page error for i = 494. Expected 1 but got -1 #19
UNEXPECTED_ERROR: Code page error for i = 495. Expected 1 but got -1 #19
UNEXPECTED_ERROR: Code page error for i = 736. Expected 2 but got -1 #19
UNEXPECTED_ERROR: Code page error for i = 737. Expected 2 but got -1 #19
...
```

Do you believe this is something worth spending time to fix or should I look for something more meaningful to work with?

#44 - 11/17/2023 07:11 AM - Greg Shah

Yes, it needs to be fixed. Such conversions should yield identical results. It seems likely that the cause is related to the multibyte processing of our versions of asc and chr.

#45 - 11/17/2023 07:41 AM - Theodoros Theodorou

Hello Marian. Greg asked me to coordinate with you that we do not do the same effort twice. Please have a look at testcases/cp/cpinternal/chr_and_asc.p and let me know if we have an overlap.

#46 - 11/17/2023 08:47 AM - Marian Edu

Theodoros Theodorou wrote:

> Hello Marian. Greg asked me to coordinate with you that we do not do the same effort twice. Please have a look at testcases/cp/cpinternal/chr_and_asc.p and let me know if we have an overlap.

Hi Theodoros, just pushed the i18n tests on testcases (rev#1514), we're still working one some changes there but what you mention here is probably what we've did in tests/i18n/TestCodepageConversions.cls. What we did was to generate 'mapping' files in 4GL with pairs of integers - the codepoint in source and codepoint in target codepage. We've found out that we're missing the four bytes characters who are by convention 'negative' numbers (unsigned int) so we are missing some mappings for those, generating the mapping files takes quite some time - there are a lot of 'empty space' there so we will ran them again at some point, right now we see if we can do some kind of optimisation in between :(

As for overlapping, as said it looks like what you try to do is more or less what we've done in that test case. It looks like you execute the procedure with different internal codepage, we're not interested by the curent codepage used in the session and always use source/target codepages in asc and

chr. Testing the fact that cpinternal is used as default if not specified is easier :)

We've had a similar approach previously but thing is whenever you use any strings those will be in cpinternal and when you save them cpstream come into play so you must know the codepage those strings were saved with otherwise when you read the file back those will be interpreted differently, the only way we think this could work is to only use numbers instead of strings for mapping. Using 'extended' characters in source code is not a good idea either as it depends on the encoding you've used when saving the file and that must be known by whoever wants to edit the file afterwards and also the 4GL compiler needs to know it - cprcode.

Other than that you should probably move your tests in tests and convert them to use ABLUnit, there is some info about this in Writing 4GL Testcases .

#47 - 11/17/2023 07:33 PM - Theodoros Theodorou

Marian Edu wrote:

> Hi Theodoros, just pushed the i18n tests on testcases (rev#1514), we're still working one some changes there but what you mention here is probably what we've did in tests/i18n/TestCodepageConversions.cls. What we did was to generate 'mapping' files in 4GL with pairs of integers - the codepoint in source and codepoint in target codepage. We've found out that we're missing the four bytes characters who are by convention 'negative' numbers (unsigned int) so we are missing some mappings for those, generating the mapping files takes quite some time - there are a lot of 'empty space' there so we will ran them again at some point, right now we see if we can do some kind of optimisation in between :(

You have done a great job, I liked the optimization removing all the -1 values and storing the data in .bin format to save space. Also, ABLUnit is more professional and elegant.

> As for overlapping, as said it looks like what you try to do is more or less what we've done in that test case. It looks like you execute the procedure with different internal codepage, we're not interested by the curent codepage used in the session and always use source/target codepages in asc and chr. Testing the fact that cpinternal is used as default if not specified is easier :)

I tried some different codepages and your test is passing so cpinternal seems to not have an effect.

> We've had a similar approach previously but thing is whenever you use any strings those will be in cpinternal and when you save them cpstream come into play so you must know the codepage those strings were saved with otherwise when you read the file back those will be interpreted differently, the only way we think this could work is to only use numbers instead of strings for mapping. Using 'extended' characters in source code is not a good idea either as it depends on the encoding you've used when saving the file and that must be known by whoever wants to edit the file afterwards and also the 4GL compiler needs to know it - cprcode.

Yes, I agree. I tried to save the actual characters but it was a mess so I saved only numbers as well in my data (.txt) files.

> Other than that you should probably move your tests in tests and convert them to use ABLUnit, there is some info about this in Writing 4GL Testcases.

Ok, you are right, although I am not sure if I should keep my test or if I should delete it because it is very similar to yours but mine is not well written.

Is this test passing with FWD?

#48 - 02/10/2024 08:13 AM - Greg Shah

- Related to Bug #7657: 8-bit character entry problem in ChUI added

#49 - 03/29/2024 06:16 AM - Greg Shah

- % Done changed from 0 to 50

I know some work has happened to explore I18N testcases. Having proper testcases will be very important.

I want to clarify the core requirement of this task. In #6431-2:

> We have a customer that needs to be able to set UTF-8 or CP963 as the CPINTERNAL. We may have some problems with doing this today with UTF-8 for which we have some really old nasty workarounds that sometimes treat UTF-8 as 8859-1 (e.g. see I18nOps.getCharsetNative()).

I've looked for that specific code and I don't see it remaining in the codebase. We have greatly improved out multi-byte support over the years. At this point I think the objective of this task is to test that support and fix any remaining issues. It may already be working properly.

Files

out.rar	903 KB	10/27/2023	Theodoros Theodorou