Base Language - Feature #1632

implement _MSG() undocumented built-in function

10/21/2012 10:38 AM - Greg Shah

Status:	Closed	**Start date:**	02/18/2013
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	Runtime Support for Server Features		
billable:	No	**vendor_id:**	GCD

Description

Subtasks:

Feature # 2011: conversion support for _MSG() built-in	**Closed**
Feature # 2012: runtime support for _MSG() built-in	**Closed**

Related issues:

Related to Base Language - Bug #2338: fix _MSG function	**Closed**	**07/10/2014**

History

#1 - 10/31/2012 01:40 PM - Greg Shah

- Target version set to Milestone 7

#2 - 02/18/2013 07:12 PM - Greg Shah

This is an undocumented built-in function. We need to determine what it does. at a high level it is used to create an error message and takes a single integer parameter. Here is a common usage pattern:

```
my-err-txt-var = "Some error message text " + STRING(INTEGER(_msg(1))).
```

Write some testcases and determine what it can do with various inputs. Try non-integer values and a proper range of integer values (large and small negative and positive values and zero).

Document how it works here.

The conversion side can be implemented in the ErrorManager.

#3 - 02/19/2013 02:20 PM - Greg Shah

From your status report:

> Tested the _msg function , results seems to not be consistent for the same value.

Please provide details. Show the examples you wrote and the output.

#4 - 02/20/2013 08:37 AM - Costin Savin

Done more testing and got a better understanding of how msg working.
It seems the _msg(int) will return the error number from a stack from newest error to oldest.

The _msg function can be used also with values < 1, and when running it for negative numbers it can return random numbers (that were also sometimes negative) this seems to be because of memory leaks.

The inconsistent part happened for _msg(0) where it sometimes had a value there (the same with the greatest index for which _msg() is not 0 -> first error number recorded) and most times it had 0 , this may be a memory leak of some sort like for negative indexes , it also happened when running external procedures non-persistent , this might also hold a clue.

msg_test.p

```
def var h as handle.
def stream s.
do on error undo,leave:
h = h:next-sibling.
end.
do on error undo,leave:
 run foo in h.
end.
do on error undo, leave:
   run proc.
end.
def var i as int.
output to msg_test.txt.
do i = -1000 to 20000:
   put string(i) string(_msg(i)) skip.
end.
output close.
procedure proc:
   do on error undo, leave:
      input stream s from OS-DIR("no_dir").
   end.
end.
```

Also the error stack is shared with external procedures :

msg_test2.p

```
define var ah as handle.
run msg_test.p persistent.
do on error undo, leave:
   ah:prev-sibling.
end.

def var i as int.
output to msg_test2.txt.
do i = 0 to 20000:
   put string(i) string (_msg(i)) skip.
end.
output close.
```

_msg can take only one parameter which must be: a number literal, an int or an int64 variable.

I've chosen ErrorManager.getErrorNumber as the conversion name for _msg , hope this is ok.

#5 - 02/20/2013 09:03 AM - Greg Shah

Please upload the msg_test.txt and msg_test2.txt output files so that I can better understand what is going on.

> I've chosen ErrorManager.getErrorNumber as the conversion name for _msg , hope this is ok.

Please go with ErrorManager.getErrorNumberAtIndex instead. It is more descriptive.

#6 - 02/20/2013 09:26 AM - Costin Savin

- File msg_test2.txt added

- File msg_test.txt added

I realized the 0 index difference happened because I mistakenly rerun the program in some cases using the character editor, instead of command line. When ran from command line _msg(0) will always be 0.
Attached the result files.

#7 - 02/20/2013 10:07 AM - Greg Shah

OK, this is more clear.

As far as the negative numbers are concerned, we won't duplicate that behavior right now. Please add this note to the javadoc:

"This implementation always will return 0 for any non-positive number (0 or negative). In the 4GL, some negative numbers will return what appears to be a random number. This is most likely a flaw in the 4GL implementation where there is improper boundary testing and an invalid index value is used to dereference memory that is not associated with the error stack. As such the behavior is non-deterministic and also is a likely security flaw. This implementation is deliberately not duplicating this bad behavior and it seems unlikely that an application would be coded to rely upon it."

I guess the last questions relate to the scoping of the stack. When does a new scope open? That should also determine when the stack gets cleared.

When you have more deeply nested scopes, I suspect that the stack will be cleared by scope (all errors related to that scope will pop off, but previous scope values will remain).

If that is the way it works, then our com/goldencode/p2j/util/PartitionedArray will probably be a good fit.

For now, I think you have done enough investigation of the features.

Please implement the conversion side and upload the proposed change.

#8 - 02/20/2013 10:26 AM - Costin Savin

- File cs_upd20130220b.zip added

Added proposed update

#9 - 02/20/2013 01:43 PM - Greg Shah

The code looks good. I only have 1 question. The javadoc says this:

```
    * @param  index
    *         The position of the stack from which we want to get the value error number.
    *         This should be greater than 1 for valid results.
```

Isn't this supposed to be "greater than 0 for valid results"? I thought that 1 is a valid input.

#10 - 02/21/2013 04:45 AM - Costin Savin

- File cs_upd20130221a.zip added

Corrected javadoc, was meant to say first: greater or equal 1, then I though greater than 0 to be better as description, but it came out neither of those..

#11 - 02/21/2013 08:00 AM - Greg Shah

Looks good. I will let you know when we can test. Only conversion regression testing will be needed. Be prepared to merge to the latest level before testing.

#12 - 02/22/2013 04:12 AM - Costin Savin

- File cs_upd20130222a.zip added

#13 - 02/22/2013 09:03 AM - Greg Shah

The update looks good. I will start testing it shortly.

#14 - 02/22/2013 11:46 AM - Costin Savin

Conversion tests passed , commited to bazaar as revision 10186

#15 - 04/12/2013 06:12 PM - Evgeny Kiselev

- File evk_upd20130412a.zip added

Attached update evk_upd20130412a.zip. Implemented getErrorNumberAtIndex method.
Also will provide testcase soon.

#16 - 04/12/2013 07:05 PM - Evgeny Kiselev

- File evk_upd20130412b.zip added

fixed error with int64

#17 - 04/12/2013 08:07 PM - Evgeny Kiselev

- File evk_upd20130412c.zip added

Attached update evk_upd20130412c.zip with testcase.

#18 - 04/13/2013 05:57 PM - Greg Shah

I am fine with the proposed changes.

I do think we need to improve the javadoc. We should explain that the getErrorNumber() methods index as a 1-based list (1 is the oldest error message and higher numbers are the more recent entries). And the getErrorNumberAtIndex() index as a 1-based stack offset, where 1 is the most recent error and higher numbers represent older error entries in the stack). Please update the javadoc accordingly, then upload that final version here.

Since this has very little or no risk to either conversion or runtime code, we will bypass regression testing. You can check in the improved update (and the testcase) into their respective bzr projects. And you can send out the update (you only have to send out the p2j code update, the testcase doesn't need to be sent via email). Follow the same format for the email as you have seen from others recently. Finally, add an entry to the history here to note the bzr revision number that corresponds with the check in.

#19 - 04/13/2013 09:03 PM - Evgeny Kiselev

- File evk_upd20130413a.zip added

Corrected javadoc according to №18 comment.

#20 - 04/14/2013 10:39 AM - Greg Shah

It looks good. You can check it in and distribute it.

#21 - 04/14/2013 05:33 PM - Evgeny Kiselev

committed evk_upd20130413a.zip in revision 10339

#22 - 04/15/2013 08:26 AM - Greg Shah

- Status changed from New to Closed

#23 - 07/07/2014 03:28 AM - Constantin Asofiei

Vadim Gindin wrote:

> I ran the *msg_test.p* and *msg_test2.p* several times and I found that there are no stack scope at all (stack scope which Greg mentioned in the note 7 of the task #1632. It seems the errors numbers stack is never gets cleared and there is no need to scope it. At least I can't find the case when it gets cleared. In the tests mentioned above there are several blocks such as several "do" blocks and calls of external and internal procedures with its own do blocks with own scopes. After execution all of them the error numbers stack contains all of the happened errors. Am I wrong?

I think you are correct; the raised errors seems to be kept in a stack, regardless of scope. More, this stack seems to be limited to 25 entries (1 to 25):

```
def var i as int.
def var n as int init 30.
def var h as handle.
```

```
repeat:
  do on error undo, leave:
  h = h:next-sibling.
  end.
  hide message no-pause .
  if i > n / 2 then leave.
  i = i + 1.
end.

repeat:
  do on stop undo, leave:
  run e.p.
  end.
  hide message no-pause .
  if i > n then leave.
  i = i + 1.
end.

output to errs.txt.
do while i >= 0:
 put string(i) string (_msg(i)) skip.
 i = i - 1.
end.
output close.
message "done".
```

which produces this result:

```
31      0
30      0
29      0
28      0
27      0
26      0
25      3140
24      3135
23      3140
22      3135
21      3140
20      3135
19      3140
18      3135
17      3140
16      293
15      293
14      293
13      293
12      293
11      293
10      293
9       293
8       293
7       293
6       293
5       293
4       293
3       293
2       293
1       293
0       3140
```

Entry 0 seems to be a random value (is not associated with the last error, and for different values of n sometimes is set to 0).

Also, as you can see, this stack is not limited to ERROR conditions: STOP condition is registered, too. Please check the QUIT and ENDKEY conditions, too.

In any case, the implementation of ErrorManager.getErrorNumberAtIndex I think should be like this:

- if the index is less than or equal to 0, or is greater than 25, return 0.
- all the raised conditions are added to a ErrorManager$WorkArea.raisedConditions fixed-size stack; the top of this stack will always represent the error number returned by the _MSG(1) call. On initialize, this stack is set to a fixed-size of 25 and is filled with zero values; as errors are pushed on the stack, elements from the bottom of the stack are discarded.

#24 - 07/07/2014 03:41 AM - Constantin Asofiei

This article states that warnings are logged too: http://knowledgebase.progress.com/articles/Article/P82729

Also, I'm not sure yet what is the best way to add elements to the ErrorManager$WorkArea.raisedConditions stack: considering that this keeps other conditions beside the ERROR condition, I think the catch (Throwable thr) in BlockManager.processBody is a good candidate: this can check if thr (or its cause, up the cause chain) is a ConditionException; if this is found, go up its cause chain and add:

- 0 if there is no NumberedException cause (you need to double-check this, it's just an assumption: what happens if QUIT. is executed and caught by an enclosing block via ON QUIT UNDO, LEAVE? Same with RETURN ERROR, which causes an ERROR condition in the calling block.
- NumberedException.getNumber, otherwise

Also, ErrorManager.recordOrThrowError needs to build proper cause chains, and not just add the last error message...

Finally: you need to check the cases which only show messages to the terminal (and no ERROR condition is raised); these are mapped in P2J by the ErrorManager.recordOrShowError calls.

#25 - 11/16/2016 11:42 AM - Greg Shah

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

Files

msg_test.txt	349 KB	02/20/2013	Costin Savin
msg_test2.txt	332 KB	02/20/2013	Costin Savin
cs_upd20130220b.zip	21.6 KB	02/20/2013	Costin Savin
cs_upd20130221a.zip	21.6 KB	02/21/2013	Costin Savin
cs_upd20130222a.zip	21.6 KB	02/22/2013	Costin Savin
evk_upd20130412a.zip	12.9 KB	04/12/2013	Evgeny Kiselev
evk_upd20130412b.zip	12.9 KB	04/12/2013	Evgeny Kiselev
evk_upd20130412c.zip	454 Bytes	04/13/2013	Evgeny Kiselev
evk_upd20130413a.zip	13 KB	04/14/2013	Evgeny Kiselev