Base Language - Bug #2338

fix _MSG function

07/10/2014 10:57 PM - Vadim Gindin

Status:	Closed	**Start date:**	07/10/2014
Priority:	Normal	**Due date:**	
Assignee:	Vadim Gindin	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	Cleanup and Stablization for Server Features		
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

Related issues:

Related to Base Language - Bug #2197: fix the error message list cleanup when...	**Closed**	**01/13/2014**	**01/21/2014**
Related to Base Language - Feature #1632: implement _MSG() undocumented built...	**Closed**	**02/18/2013**	

History

#1 - 07/10/2014 11:58 PM - Vadim Gindin

From #1632 _MSG() is built-in undocumented function.

Constantin Asofiei wrote:

> 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):
> [...] (msg25.p)
> which produces this result:
> [...] (result.txt)
> 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.

#2 - 07/11/2014 12:13 AM - Vadim Gindin

Constantin Asofiei wrote:

> 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.

#3 - 07/11/2014 02:02 AM - Vadim Gindin

- File vig_upd20140711a.zip added

Here is the first version of implementation. Now I'm working on QUIT condition and recordOrShowError (errors without sign of error).

Questions.
1. It seems there are some trick in handle (see my change). There are 2 errors happenning and the second code is -1. Don't you remember why? I fixed it for the current implementation but I'm not sure if it is ok.
2. I added collection of errors numbers to NumberedException. Not so good. But I thought it is better suits to the sense of situation. It is more list of errors (during one statement, like in handle) than chain of causes.

What do you think?

#4 - 07/11/2014 07:50 AM - Greg Shah

- Subject changed from implement _MSG function to fix _MSG function

- Status changed from New to WIP

#5 - 07/11/2014 10:04 AM - Constantin Asofiei

Vadim, the logic looks good. The only part I'm not comfortable with is the NumberedException changes: you are keeping a single message, but the error numbers can be more than one.

I think is best to chain the NumberException's (via their cause). Also, you need to check if the errors which are shown at the terminal (but not raised) are recorded, too; P2J treats these via the ErrorManager.recordOrShowError API. A simple example is the add-super-procedure test from #2197 note 18, without the no-error clause. When looking into the chain, start from the Condition's cause and keep retrieveing numbered exceptions as long

> 1. It seems there are some trick in handle (see my change). There are 2 errors happenning and the second code is -1. Don't you remember why? I fixed it for the current implementation but I'm not sure if it is ok.

I think you caught a bug, the second case was net assigning the error number to the correct value. int code2 = -1 was used to not leave the var uninitialized. On a side note, you need to merge the handle.java file.

To resume, what else needs to be tested:

- QUIT/STOP/ENDKEY conditions
- conditions with multiple messages (to ensure they are recorded right)
- warnings
- cases when a message is shown at the terminal, but the ERROR condition is not raised. In NO-ERROR terms, this means that the ERROR-STATUS:NUM-MESSAGES/GET-MESSAGE contains data, but the ERROR-STATUS:ERROR flag is not set.

#6 - 08/12/2014 04:07 PM - Vadim Gindin

Constantin Asofiei wrote:
..

> - QUIT/STOP/ENDKEY conditions

I didn't find a way to test QUIT and ENDKEY conditions with errors. I.e. I didn't find a way to throw an error with QUIT or ENDKEY condition. STOP condition was already tested in msg25.p

> - conditions with multiple messages (to ensure they are recorded right)

Already tested in msg25.p. h:next-sibling produces 2 errors 3135 and 3140. Both a recorded

> - warnings
> - cases when a message is shown at the terminal, but the ERROR condition is not raised. In NO-ERROR terms, this means that the ERROR-STATUS:NUM-MESSAGES/GET-MESSAGE contains data, but the ERROR-STATUS:ERROR flag is not set.

I modified your test with add-super-procedure and found that such messages are also being recorded by _MSG function. Are such messages named warnings? If warnings are something other could you describe it more in detail.

#7 - 08/12/2014 04:21 PM - Constantin Asofiei

Vadim Gindin wrote:

> Constantin Asofiei wrote:
> ..
>
> - QUIT/STOP/ENDKEY conditions
>
> I didn't find a way to test QUIT and ENDKEY conditions with errors. I.e. I didn't find a way to throw an error with QUIT or ENDKEY condition. STOP condition was already tested in msg25.p

QUIT condition I don't think can be raised without explicitly calling QUIT.. About ENDKEY - this usually is raised when F4 is pressed, but I don't think a message is shown on terminal. Search for EndConditionException references throughout the P2J project and you will find these:

```
Stream.java (3 matches)
1,510: throw new EndConditionException("EOF");
1,903: throw new EndConditionException("EOF");
5,338: throw new EndConditionException("EOF");
InMemoryLockManager.java
2,047: throw new EndConditionException(exc);
BufferValidator.java
83: throw new EndConditionException(e);
```

and these (as QueryOffEndException is a sub-class of EndConditionException):

```
AdaptiveQuery.java
2,346: throw new QueryOffEndException("Missing join record");
CompoundQuery.java (2 matches)
1,513: throw new QueryOffEndException("",
1,882: throw new QueryOffEndException("",
RecordBuffer.java
6,843: throw new QueryOffEndException("", new QueryOffEndListener[] { offEndListener });
```

Try to create a test which reaches a few of these code paths (when a condition is raised) and see if it gets recorded in _MSG.

- cases when a message is shown at the terminal, but the ERROR condition is not raised. In NO-ERROR terms, this means that the ERROR-STATUS:NUM-MESSAGES/GET-MESSAGE contains data, but the ERROR-STATUS:ERROR flag is not set.

> I modified your test with add-super-procedure and found that such messages are also being recorded by _MSG function. Are such messages named warnings? If warnings are something other could you describe it more in detail.

Yes, I think so. The way to prove it is to disable the warnings:

```
session:suppress-warnings = true.

this-procedure:add-super-procedure(this-procedure, search-self).
this-procedure:add-super-procedure(this-procedure, search-target).

message _msg(1).
```

This doesn't show the message on screen, but the _msg records it. More details which seem to confirm this I found here:
http://knowledgebase.progress.com/articles/Article/P82729

#8 - 09/07/2014 02:45 PM - Vadim Gindin

I tried to test these mysterious cases. It is not easy. Here are some findings.

1. Stream.readBlock() method, that can raise EndConditionException.
While trying to reproduce the procedure, converted java app of wich calls this method I made the test */errlist/test_msg_endkey_stream.p*
Its behavior differs from it's JAVA app version.
The goal is to create a buffer (memptr) of specified size and try to copy some data to it from a file, that has different size than a buffer. Here are possible several ways. I'm writing behavior and the difference.

1a. size(memptr) < size(file). For exapmle size(memptr) = 5 and size(file) = 23.
Progress reads all file and dynamically increases memptr size to needed value i.e. from 5 to 23
Java impl only reads size(memptr) bytes from a file, i.e. 5
1b. size(memptr) > size(file). For exapmle size(memptr) = 35 and size(file) = 23.
Both sides works equal except the one border case when the size of a file is 0.
In that case Progress works correctly without errors but Java implementation raises EndConditionException, that is recorder to _MSG array.

Conclusion. My test does not raised an error in Progress so I couldn't investigate ENDKEY behavior in this case, but I found the difference in common behavior that probably should be corrected in other task. What to do with that?

2. BufferValidator.validate() method, that can raise EndConditionException.
I wrote */errlist/test_msg_endkey_table.p*, that creates temp table with 2 possible constraints (format constraint 1-digit number and UNIQUE constraint) and tries to fill it with incorrect data.

2a. format constraint 1-digit number.
While trying to validate number 10 there are an error without error condition is shown and recorded to _MSG array. At the same time Java implementation displays the error in NumberType.genCannotBeDisplayed() method and does not raise an ErrorConditionException and so it can't be caught and recorded to _MSG array with the current implementation.

2b. Unique constraint violation.
It raises ValidationException in DMOValidator instead of BufferValidator as I expected. This exception after that is being wrapped in ErrorConditionException and stops procedure. Java implementation works in the same way. Error is recorded to _MSG. but I don't have possibility to read the value from there because procedure ends right after that error.

Conclusion.
In case of not raised errors and warnings I should probably change ErrorManager methods, that can show an error without raising exception to add _MSG processing. What do you think?
I didn't test BufferValidator as it was needed. To do that I probably should add some user defined constraint. For example, all column values should be in the interval [5, 10]. Am I correctly understand BufferValidator mission?

#9 - 09/07/2014 02:50 PM - Vadim Gindin

I forgot to add for the case №1 "Stream", that STRING in Progress successfully outputs string representation of its value, but the Java implementation always outputs "". See memptr.toString(). So it probably should also be corrected.

#10 - 09/08/2014 02:09 PM - Greg Shah

> 1a. size(memptr) < size(file). For exapmle size(memptr) = 5 and size(file) = 23.
> Progress reads all file and dynamically increases memptr size to needed value i.e. from 5 to 23
> Java impl only reads size(memptr) bytes from a file, i.e. 5
> 1b. size(memptr) > size(file). For exapmle size(memptr) = 35 and size(file) = 23.
> Both sides works equal except the one border case when the size of a file is 0.
> In that case Progress works correctly without errors but Java implementation raises EndConditionException, that is recorder to _MSG array.

Our implementation of COPY-LOB is known to be incomplete. Please edit LargeObjectOps.readFromFile() and add a TODO comment that describes these problems.

You don't have to fix these issues.

> that STRING in Progress successfully outputs string representation of its value, but the Java implementation always outputs "". See memptr.toString(). So it probably should also be corrected.

Yes, you're right. It seems to treat any contents as a null terminated string. I have added task #2386 for this. You don't have to fix it.

I will let Constantin reply regarding your other questions.

#11 - 09/09/2014 06:57 AM - Constantin Asofiei

Vadim Gindin wrote:

> 2a. format constraint 1-digit number.
> While trying to validate number 10 there are an error without error condition is shown and recorded to _MSG array. At the same time Java implementation displays the error in NumberType.genCannotBeDisplayed() method and does not raise an ErrorConditionException and so it can't be caught and recorded to _MSG array with the current implementation.

This can also be duplicated via:

```
def var i as int init 10 format "9".

session:suppress-warnings = true.
display i with frame f1.
message _msg(1).
```

Strangely, this shows that the "74" message is not a warning... although no condition is raised, I think in their runtime, 4GL might "consume" certain raised conditions in certain cases. If you add a NO-ERROR to the display statement, the ERROR-STATUS:ERROR flag remains no, but the ERROR-STATUS:GET-MESSAGE contains the * Value 10 cannot be displayed using 9. (74). So our approach seems correct.

> 2b. Unique constraint violation.
> It raises ValidationException in DMOValidator instead of BufferValidator as I expected. This exception after that is being wrapped in ErrorConditionException and stops procedure. Java implementation works in the same way. Error is recorded to _MSG. but I don't have possibility to read the value from there because procedure ends right after that error.

You can enclose the code which raises an ERROR condition in a DO ON ERROR UNDO, LEAVE: ... END. block and check the _MSG value after this block.

> In case of not raised errors and warnings I should probably change ErrorManager methods, that can show an error without raising exception to add _MSG processing. What do you think?

Yes, I think this is the correct way.

> I didn't test BufferValidator as it was needed. To do that I probably should add some user defined constraint. For example, all column values should be in the interval [5, 10]. Am I correctly understand BufferValidator mission?

Yes, you are correct. But I think BufferValidator works only with permanent tables. In p2j_test.df, there is a restriction in the book table, so that the title Bogus Programming can not be deleted. Using a test like this (a book with this title is required):

```
def var i as int init 10.
display i format "9".

message _msg(1). /* ensure the 1st slot is occupied by something known. */

find first book where book-title = "Bogus Programming".
do on error undo, leave:
   delete book.
end.

message error-status:error error-status:num-messages error-status:get-message(1) error-status:get-number(1).

message _msg(1). /* this shows "74" */
```

I think it shows that:

1. if a NO-ERROR is added to DELETE book, then the ERROR-STATUS sets the error flag, the number to 0, and the error message to the valmsg value - so BufferValidator seems to work OK.
2. but if the NO-ERROR clause is not used, then _msg still shows "74" instead of "0" - not sure how they do this, they might either not call at all the code which adds values to the _MSG stack or maybe they are ignoring error-numbers which are set to 0.

#12 - 09/09/2014 12:49 PM - Vadim Gindin

Constantin Asofiei wrote:

> I think it shows that:
>
> 1. if a NO-ERROR is added to DELETE book, then the ERROR-STATUS sets the error flag, the number to 0, and the error message to the valmsg value - so BufferValidator seems to work OK.
> 2. but if the NO-ERROR clause is not used, then _msg still shows "74" instead of "0" - not sure how they do this, they might either not call at all the code which adds values to the _MSG stack or maybe they are ignoring error-numbers which are set to 0.

I'm not sure, that I understood correctly. I expected that 0 is the default value that exists in _MSG array before any error will happen. I.e. _MSG(1) = 0 - right at start of procedure. As I can see here are two conclusions.
1. With NO-ERROR the error number wasn't recorded to _MSG array. That is little unexpected.
2. Without NO-ERROR the error number was recorded as it goes when any other error happens. That is expected.
Is this the same as your conclusions?

Question.
Are there a way to reproduce this situation without real database? You wrote that BufferValidator works only with real database. Why? I would want to avoid environment setup with database in this case.

#13 - 09/09/2014 01:02 PM - Constantin Asofiei

Vadim Gindin wrote:

> I'm not sure, that I understood correctly. I expected that 0 is the default value that exists in _MSG array before any error will happen. I.e. _MSG(1) = 0 - right at start of procedure. As I can see here are two conclusions.
> 1. With NO-ERROR the error number wasn't recorded to _MSG array. That is little unexpected.

Please correct me, but as I recall NO-ERROR prevents data to be recorded into _MSG, right?

> 2. Without NO-ERROR the error number was recorded as it goes when any other error happens. That is expected.

No, your conclusion is incorrect. Without NO-ERROR, the _MSG doesn't get changed. In my example, the _MSG(1) was set to "74" by the display i format "9". code. After the DELETE book was executed, the _MSG(1) remains unchanged (thus nothing gets pushed onto the stack).

> Are there a way to reproduce this situation without real database? You wrote that BufferValidator works only with real database. Why? I would want to avoid environment setup with database in this case.

You need a real DB because in 4GL VALEXP/VALMSG clauses can be set only for physical tables. To add a 4GL DB, connect to the Data Dictionary, create a new DB and then import the p2j_test.df file. To start from the command line, use mpro -db <path to db folder> -p <program.p>. For P2J, details can be found in the documentation (details in one of the chapters in the Runtime installation, administration... book I think should be enough).

#14 - 09/09/2014 01:21 PM - Vadim Gindin

1. With NO-ERROR the error number wasn't recorded to _MSG array. That is little unexpected.

> Please correct me, but as I recall NO-ERROR prevents data to be recorded into _MSG, right?

Sorry, I forgot it myself. You're right NO-ERROR prevents data to be recorded into _MSG.

#15 - 09/09/2014 04:07 PM - Vadim Gindin

I ran your test and in both cases (with or without NO-ERROR) it behaved in the same way. _MSG(1) = 74. The error being raised in delete book is not getting in _MSG array.. Strange...

#16 - 09/09/2014 04:23 PM - Constantin Asofiei

Vadim Gindin wrote:

> I ran your test and in both cases (with or without NO-ERROR) it behaved in the same way. _MSG(1) = 74. The error being raised in delete book is not getting in _MSG array.. Strange...

OK, for now I think is safe to do not push to the _MSG stack when the error number is 0.

#17 - 09/15/2014 08:19 AM - Vadim Gindin

- File vig_upd20140915a.zip added

Added implementation for warnings to ErrorManager methods.

Questions.
1. There are 2 methods ErrorManager.isSilent() and ErrorManager.isSilentError(). They calls 2 different objects methods: ContextContainer - local context and RemoteErrorData - da. Are these methods always return the same value. I.e. is isSilent() = isSilentError()? If not - which one of them to use better in the method addRaisedExceptions?
2. How to set headless mode. (see isHeadless() method). Do I need to check this case for my task?

#18 - 09/17/2014 04:10 AM - Constantin Asofiei

Vadim Gindin wrote:

> Added implementation for warnings to ErrorManager methods.
>
> Questions.
> 1. There are 2 methods ErrorManager.isSilent() and ErrorManager.isSilentError(). They calls 2 different objects methods: ContextContainer - local context and RemoteErrorData - da. Are these methods always return the same value. I.e. is isSilent() = isSilentError()?

Although the isSilent and isSilentError return the same state, they are different:

1. isSilent can be called from both server and client-side; when called from client-side, it interrogates the server-side (where the state of this flag is really kept).
2. isSilentError can be called only from server-side, as it accesses the state directly.

> If not - which one of them to use better in the method addRaisedExceptions?

I think you need to use isSilent, as it can be called from both client and server-side safely.

Do you have anything else to do?

Please remove the TODO from handle.java:2047

> 2. How to set headless mode. (see isHeadless() method). Do I need to check this case for my task?

Headless mode is used only when the ErrorManager is used from PL/Java - so I don't think you need to worry about it.

#19 - 09/17/2014 09:45 AM - Vadim Gindin

This is a final update. Can I run regression testing?

#20 - 09/17/2014 09:46 AM - Constantin Asofiei

Vadim Gindin wrote:

> This is a final update. Can I run regression testing?

If your standalone tests all pass (including the cases we've discussed here), then you can go ahead with regression testing.

Please remove the TODOs left in the code and post the final update here, for a final review.

#21 - 09/17/2014 10:45 AM - Vadim Gindin

- File vig_upd20140917a.zip added

Yes, I checked all tests. Here is the final update.

#22 - 09/19/2014 06:04 AM - Vadim Gindin

The update vig_upd20140917a.zip has passed regression testing. Can I commit it?

#23 - 09/19/2014 09:47 AM - Constantin Asofiei

Vadim Gindin wrote:

> The update vig_upd20140917a.zip has passed regression testing. Can I commit it?

Yes, go ahead and release it.

#24 - 09/19/2014 10:35 AM - Vadim Gindin

committed to revision 10614

#25 - 09/19/2014 12:30 PM - Greg Shah

- % Done changed from 0 to 100

- Status changed from WIP to Closed

- Target version set to Milestone 11

#26 - 11/16/2016 12:06 PM - Greg Shah

- Target version changed from Milestone 11 to Cleanup and Stablization for Server Features

Files

vig_upd20140711a.zip	60.7 KB	07/11/2014	Vadim Gindin
vig_upd20140915a.zip	124 KB	09/15/2014	Vadim Gindin
vig_upd20140917a.zip	124 KB	09/17/2014	Vadim Gindin