Base Language - Bug #2659

memptr in internal procedure address duplication

08/24/2015 04:22 PM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

Related issues:

Related to Base Language - Feature #1635: implement MEMPTR/RAW support	**Closed**	**01/19/2013**	**05/03/2013**

History

#1 - 08/24/2015 04:48 PM - Greg Shah

The following code can demonstrate a disturbing behavior that occurs in the 4GL:

```
def var ptr as int64 extent 100.

procedure proc0.
   def input param idx as int.

   def var m as memptr.
   set-size(m) = 1024.
   put-int64(m, 1) = idx.
   ptr[idx] = get-pointer-value(m).
end.

def var i as int.
def var m3 as memptr.
def var i2 as int64.

do i = 1 to 100:
  run proc0(input i).
  set-pointer-value(m3) = ptr[i].
  i2 = get-int64(m3, 1).
  if i2 <> i then message "ptr " + string(i) + " has unexpected value " + string(i2).
end.

def var p1 as int64.
p1 = ptr[1].
do i = 2 to 100:
  if p1 <> ptr[i] then do: p1 = ?. leave. end.
end.

if p1 = ? then message "all pointers must be equal!".

def var m2 as memptr.
set-pointer-value(m2) = ptr[1].
def var i1 as int64.
i1 = get-int64(m2, 1).

if i1 <> 100 then message "ptr 1 has value" i1 "instead of 100!".
```

It is not clear if exiting the internal procedure deallocates the memptr that was created inside it, because the memory itself is likely still valid (cached by the CRT). If the exit of the internal procedure does not deallocate the memptr, then this is a bug in the 4GL since the next call to the internal procedure will return the same exact address.

This needs to be investigated further to see if some kind of automatic deallocation happens or not. Either way, we don't plan to implement the address duplication "feature", unless we find real customer code that relies upon it and which cannot be easily changed. The primary concern is whether we are missing some automatic deallocation in our implementation.