OWrite spell check endless loop problem

In version 1.62 of OWrite we had to change the way OWrite implements the interactive spell check. We have found that in some complex libraries when used with Studio 4.1b2, OWrite would cause an endless loop when prompting the user to correct a misspelled word via the spell checker window. OWrite version 1.61 relies on the Omnis “Enter data” command to block both the 4GL method and the C++ call stack. Studio 4.1b2 does not appear to block the C++ call stack in some circumstances. We do not know what the trigger for this failure is and as a result we were forced to change the way OWrite handles interactive spell checking.

How it used to work

When the user chooses to spell check an OWrite field, the main spell checker object would call the OWrite method $spell(kWriSpllCheckSelection) or $spell(kWriSpllCheckAll). This method loops through the OWrite data looking for incorrectly spelled words. When a word is found, OWrite would then call the 4GL method $owritespell of the OWrite spell session object. This method typically opens the spell checker window and blocks execution with an Enter data command.

How it works now

In version 1.62 of OWrite, $spell has two new parameters, “Context ID” and “Result Code”. A call to $spell will now look like this.

```
OWriteRef.$spell(kWriSpllCheckAll,ContextID,ResultCode)
```

When calling $spell for the first time in a session for an individual OWrite field, ContextID must be zero.

When OWrite finds an incorrectly spelled word, it returns control to the caller. On return, ContextID will contain a valid ID and ResultCode will be set to one of the spell checker result code constants.

The spell checker window’s $pompt method requires the result code to prompt the user with the appropriate details. Now that the external is no longer on the call stack it does not matter if Omnis fails to block the C++ call stack.

When the user has corrected the word, you call $spell again and again until ContextID is set to zero or ResultCode is set to kSpllRsltOK.

To fully implement this fix you must use version 1.62 of the OWrite component and make the following changes to your libraries.

Change the top section dealing with OWrite fields of the method oSpellChecker.$checkfield in the OSpell2 library. If you have not made any changes to this method, you may copy this method from the new examples.

```
Do method initialize
If pTheField.$componentlib='OWrite' ;; OWrite will handle checking directly - OWRITE
  Repeat     ;; mpm11NOV05_2 begins
    If pIsCheckWindow
      Do pTheField.$spell(kWriSpllCheckAll,lOWriteContext,lOWriteResult)
    Else
      Do pTheField.$spell(kWriSpllCheckSelection,lOWriteContext,lOWriteResult)
    End If
    If lOWriteContext
      Do method $promptspellwindow (pTheField.$spellsession,lOWriteResult,pTheField)
      Calculate pCanceled as ivSpellWind.$iscancel
      If isnull(pCanceled)
        Calculate pCanceled as kFalse
      End If
    End If
  Until pCanceled|(lOWriteContext=0)     ;; mpm11NOV05_2 ends
  If pCanceled     ;; inform OWrite of the cancel
    Do pTheField.$spell(kWriSpllCancel,lOWriteContext)
  End If
  If pCloseWindow
    Do method $endcheck (not(pCanceled))
  End If
Else
...existing code...
```

You must add the following custom notation to each of your OWrite window fields.

```
$spellsession

;; return a reference to your OWrite spell session object, i.e.
Quit method ivOWriteSpellSession.$ref
```