Uncompressing Blobs on the long_blob table

The following example checks to see if a blob is compressed. If the blob is compressed it is uncompressed and the rtf is stripped out. If the blob is not compressed it is used as is.

```
DROP PROGRAM 1_test_long_blob GO
CREATE PROGRAM 1_test_long_blob

PROMPT          "Output to File/Printer/MINE" = MINE
WITH   OUTDEV


declare OCFCOMP_VAR = f8 with Constant(uar_get_code_by("MEANING",120,"OCFCOMP")),protect
declare NOCOMP_VAR = f8 with Constant(uar_get_code_by("MEANING",120,"NOCOMP")),protect
declare  BlobOut = vc
declare  BlobNoRTF =  vc
declare bsize = i4

SELECT INTO $OUTDEV
        C.CE_EVENT_NOTE_ID,
        C.COMPRESSION_CD,
        C_COMPRESSION_DISP = UAR_GET_CODE_DISPLAY( C.COMPRESSION_CD ),
        L.LONG_BLOB,
        L.PARENT_ENTITY_NAME,
        lenblob = size( L.LONG_BLOB )

FROM
        CE_EVENT_NOTE  C,
        LONG_BLOB  L

Plan c where C.COMPRESSION_CD IN (NOCOMP_VAR, OCFCOMP_VAR)
JOIN l where C.CE_EVENT_NOTE_ID  =   L.PARENT_ENTITY_ID AND
   L.PARENT_ENTITY_NAME = "CE_EVENT_NOTE"

Head Report
        m_NumLines = 0
%I cclsource:vcclrtf.inc

Detail
        if ((ROW + 3) >= maxrow)  break endif
        blobout = notrim(fillstring(32768," "))
        blobnortf = notrim(fillstring(32768," "))

        if(c.compression_cd = ocfcomp_var)
          ;use a variable to get the actual uncompressed size
          uncompsize = 0
          ;use uar_ocf_uncompress to uncompress the blob
    ;the uncompressed blob is assigned to the variable blobout
           blob_un = UAR_OCF_UNCOMPRESS
                   (l.long_blob, lenblob,
                    BLOBOUT, SIZE( BLOBOUT ), uncompsize)

          ;use uar_rtf2 to strip the rtf from the blob
               stat = uar_rtf2(blobout,uncompsize,
                           blobnortf,size(blobnortf),bsize,0)
```

```
 ;set blobnortf to actual size
  blobnortf = substring(1,bsize,blobnortf)
 else
  blobnortf = l.long_blob
endif
COL 5  C.CE_EVENT_NOTE_ID
COL 29  C_COMPRESSION_DISP
COL 83  L.PARENT_ENTITY_NAME
ROW + 1
;use the print on multiple lines option in VE to wrap the blob.
CALL cclrtf_print( 0, 13, 80, blobnortf, size(blobnortf), 1 )
ROW + 1

WITH MAXREC = 100, NOHEADING, FORMAT= VARIABLE

END
GO
```

Uncompressing Blobs

The following example uncompress a blob, strips out the rtf and displays the blob contents in ascii format:

```
drop program ccl_blob go
create program ccl_blob

;execute cclseclogin

SET OcfCD = 0.0
Set stat = uar_get_meaning_by_codeset(120,"OCFCOMP",1,OcfCD)

set BlobOut= fillstring( 32768, ' ' )
set BlobNoRTF =  fillstring( 32768, ' ' )
set bsize = 0

select
        BlobIn  = trim(Cb.BLOB_CONTENTS),
        textlen = textlen(cb.blob_contents)
from
     ce_blob cb

where
        Cb.COMPRESSION_CD = OcfCD

Head Report

/* The following subroutine wraps ascii text for display.
This subroutine WILL NOT WORK with postscript reports. This subroutine can be used
in CCL programs to wrap textual fields in report writer.
In Visual Explorer, it can be included using %I in the Subroutines
section of the Report menu and then called in the report. A better method for
wrapping the text in VE would be to use the print across multiple lines option.  To
see this option place a field on the report writer layout grid and then double
click it.  The print across multiple lines option includes the
```

```
cclsource:vcclrtf.inc file that is shipped with VE and then calls the cclrtf_print
subroutine to print the text on multiple lines.
*/

SUBROUTINE CCL_text_wrap(X, Y, Z)  ;name of the subroutine is CCL_TEXT_wrap

;The parameters X, Y, and Z will be passed from the program when this
;subroutine is called.
;X = column where item is placed on report
;Y = the length of the text string before it wraps
;Z = the textual field that is being placed on the report

;initialize variables
        eol = SIZE(TRIM(Z),1)   ;finds total length of trimmed textual field
        bseg = 1
        eseg = 1
        line = SUBSTRING(bseg, eol, Z)

        while(eseg <= eol )
                bseg = eseg
                eseg = eseg + y
                ;if there is a space in the segment, break on the space
                if(findstring(" ",substring(bseg,eseg-bseg,line))>0)
                     while(substring(eseg -1,1,line) != " " and eseg !=bseg)
                                eseg = eseg -1
                     endwhile
                     segment = substring(bseg,(eseg - bseg) -1, z)
                else
                     segment = substring(bseg,(eseg - bseg), z)
                endif
                col x call print(substring(1,y,segment))
                row +1
        endwhile
END


        cntr = 0

Detail
        cnt  = 1
        cntr = cntr +1
        col  0 "Record:" , cntr
        col +2 "Event ID:" ,cb.event_id
        ;use uar_ocf_uncompress to uncompress the blob
        ;the uncompressed blob is assigned to the variable blobout
        blob_un = UAR_OCF_UNCOMPRESS
                (cb.blob_contents, textlen,
                 BLOBOUT, SIZE( BLOBOUT ), 32768)

        ;use uar_rtf2 to strip the rtf from the blob
        stat = uar_rtf2(blobout,size(blobout),
                        blobnortf,size(blobnortf),bsize,0)

        col +1 call ccl_text_wrap(col, 100, blobnortf)
```

```
        row +1

WITH    MAXREC = 20,
        MAXCOL = 32000,
        NOHEADING,
        FORMAT = VARIABLE

end
go

**NOTE if the blob appears to be overlaying, reinitialize BlobOut and BlobNoRTF
 BlobOut= fillstring( 32768, ' ' )
 BlobNoRTF =  fillstring( 32768, ' ' )
```

Stripping RTF out of Blobs

The uar_rtf and uar_rtf2 routines can be used to strip the rtf formatting commands out of a blob. The following example shows how to uncompress the blob, qualify on the blob, and strip out the rtf commands.

```
DROP PROGRAM blobtest GO
CREATE PROGRAM blobtest

execute cclseclogin

SET OcfCD = 0.0
Set stat = uar_get_meaning_by_codeset(120,"OCFCOMP",1,OcfCD)

set BlobOut= fillstring( 32768, ' ' )
set BlobNoRTF =  fillstring( 32768, ' ' )
set bsize = 0

SELECT
   tlen = textlen(c.blob_contents),
   BlobIn = trim(C.BLOB_CONTENTS)
FROM    CE_BLOB  C, dummyt d
PLAN C  WHERE  C.COMPRESSION_CD = OcfCD
join d where
            UAR_OCF_UNCOMPRESS(c.blob_contents,textlen(c.blob_contents),
            BLOBOUT, SIZE( BLOBOUT ), 32768) >= 0
            and blobout = "*chest*"
            ;uar_ocf_uncompress will set blobout equal
            ;to the uncompressed blob
            ;
            ;the dummyt table must be used to qualify
            ;on the string value in the blob
            ;i.e. "and blobout = "*chest*"

Head Report

;this routine wraps text for display
%i cclsource:CCL_text_WRAP.inc
        cntr = 0
```

```
Detail
        cnt  = 1
        cntr = cntr +1
        col 0 "Record:", cntr
        ;use uar_rtf2 to strip the rtf from the blob
        stat = uar_rtf2(blobout,size(blobout),
                        blobnortf,size(blobnortf),bsize,0)

        col +1 call ccl_text_wrap(col, 100, blobnortf)
        row +1

WITH    MAXREC = 20,
        MAXCOL = 32000,
        NOHEADING,
        FORMAT = VARIABLE


END
GO
blobtest go


/*
;The above program calls the ccl_text_wrap subroutine.
;This subroutine WILL NOT WORK with postscript reports.
;The subroutine is created by including the
;cclsource:ccl_test_wrap.inc file in the head reportsection of the
;select command.  This subroutine can be used in CCL programs to wrap
;textual fields in reportwriter.  The source code contained in the
;cclsource:ccl_test_wrap.inc file is shown below.
;A better method for wrapping the text in VE would be to use the print
;across multiple lines option.  To see this option place a field on the
;report writer layout grid and then double click it.  The print across
;multiple lines option includes the cclsource:vcclrtf.inc file that is
;shipped with VE and then calls the cclrtf_print subroutine to print the
;text on multiple lines.

SUBROUTINE CCL_text_wrap(X, Y, Z)  ;name of the subroutine is
                                   ;CCL_TEXT_wrap

;The parameters X, Y, and Z will be passed from the program when this
;subroutine is called.
;X = column where item is placed on report
;Y = the length of the text string before it wraps
;Z = the textual field that is being placed on the report

;initialize variables
        eol = SIZE(TRIM(Z),1)   ;finds total length of trimmed textual field
        bseg = 1
        eseg = 1
        line = SUBSTRING(bseg, eol, Z)

        while(eseg <= eol )
                bseg = eseg
```

```
                eseg = eseg + y
                ;if there is a space in the segment, break on the space
                if(findstring(" ",substring(bseg,eseg-bseg,line))>0)
                   while(substring(eseg -1,1,line)!=" " and eseg!=bseg)
                                eseg = eseg -1
                   endwhile
                   segment = substring(bseg,(eseg - bseg) -1, z)
                else
                     segment = substring(bseg,(eseg - bseg), z)
                endif
                col x call print(substring(1,y,segment))
                row +1
        endwhile
END
*/
```

Qualifying on Blob fields

Here is an example of restricting the blob in the where clause, first the blob is uncompressed before the search is applied. Also, the blobout variable is declared before the select. Qualifying on blobs is inefficient so other qualifications must be used to limit the number of blob that are read.

```
DROP PROGRAM blobtest GO
CREATE PROGRAM blobtest

SET OcfCD = 0.0
Set stat = uar_get_meaning_by_codeset(120,”OCFCOMP”,1,OcfCD)
set BlobOut= fillstring( 32768, ' ' )

SELECT
   tlen = textlen(c.blob_contents),
   BlobIn = trim(C.BLOB_CONTENTS)
FROM    CE_BLOB  C, dummyt d
PLAN C  WHERE  C.COMPRESSION_CD = OcfCD
join d where  ASSIGN(BlobOut, fillstring( 32768, ' ' ))
;CLEARS OUT THIS VARIABLE FOR THE NEXT RECORD CWC 02/09/05
      AND UAR_OCF_UNCOMPRESS(c.blob_contents,textlen(c.blob_contents), BLOBOUT,
      SIZE( BLOBOUT ), 32768) >= 0
      and blobout = "*chest*"
Head Report
        ;BlobOut= fillstring( 32768, ' ' )
        cntr = 0

Detail
        cnt  = 1
        cntr = cntr +1
        col 0 "Record:", cntr
        bsize = size(trim(blobout))
        col +2 bsize
        row +1
        while(cnt < bsize )
                line = substring(cnt, 100,blobout)
                col 25 line
                row +1
```

```
                cnt = cnt +100
        endwhile

WITH    MAXREC = 20,
        MAXCOL = 32000,
        NOHEADING,
        FORMAT = VARIABLE

END
GO
```

If the blob is uncompressed you can qualify on it by joining to the dummyt table.

```
DROP PROGRAM ve_blobtest GO
CREATE PROGRAM ve_blobtest

SELECT  BlobOut = trim(  C.BLOB_CONTENTS )

FROM    CE_BLOB  C, dummyt d

PLAN C
Join d where  check(c.blob_contents) = "*chest*"

Head Report
        BlobOut= fillstring( 32768, ' ' )
        cntr = 0

Detail
        cnt  = 1
        cntr = cntr +1
        col 0 "Record:", cntr

      bsize = size(trim(blobout))
        col +2 bsize
        row +1
        while(cnt < bsize )
                line = substring(cnt, 100,blobout)
                col 25 line
                row +1
                cnt = cnt +100
        endwhile


WITH    MAXREC = 20,
        MAXCOL = 32000,
        NOHEADING,
        FORMAT = VARIABLE

END
```

```
GO
ve_blobtest GO
```

Working with blobs on the ce_blob table

```
/*
When blobs are stored on the ce_blob table, one actual blob may be broken up into multiple rows on
the table.  Each row will contain upto a 32k segment of the blob.  The text string "ocf_blob" will
be appended to each of the rows. All of the rows that have the same event_id where the current
date and time is between the valid_from_dt_tm and the valid_until_dt_tm will make up one actual blob.
The blob_seq_num field is used to determine the order of the individual blob segements.

The following example program selects the rows for a specific event_id, concats them into a single
variable, uncompresses the blob and writes it to a file.
*/

drop program 1_multi_record_blob go
create program 1_multi_record_blob

;***** declare blob working variables
declare good_blob = vc
declare print_blob = vc
declare outbuf = c32768
declare blobout = vc

declare retlen = i4
declare offset = i4
declare newsize = i4
declare finlen = i4
declare xlen=i4

;for testing set event_id to a specific event_id
;a real program would probably join to the ce_blob table using the event_id
declare event_id = f8 with constant(398851264.0)
declare event_id_str = c20 with constant(cnvtstring(event_id))
declare file_name = vc with constant(build("1_", event_id_str, ".RTF"))

select into value(file_name)
        cb.event_id,
        cb.blob_seq_num,
        cb.BLOB_LENGTH,
        cb.VALID_FROM_DT_TM,
        cb.VALID_UNTIL_DT_TM

from
        ce_blob  cb

where  cb.event_id = event_id
        ;some processes appear to write new rows when the blob is updated
        ;the old rows will have a valid_until_dt_tm before the current date/time
        ;the following qualification gets only the valid rows
```

```
   and  cb.VALID_FROM_DT_TM < cnvtdatetime(curdate,curtime3)
   and  cb.VALID_UNTIL_DT_TM > cnvtdatetime(curdate,curtime3)

order by
          cb.event_id,
          cb.blob_seq_num

head cb.event_id
   ;****** initialize blobout to a size that will be large enough to hold the full uncompressed blob
   ;****** initialize space for the 32k segments
   for (x = 1 to (cb.blob_length/32768) )
             blobout = notrim(concat(notrim(blobout),notrim(fillstring(32768, " "))))
          endfor
          finlen = mod(cb.blob_length,32768)
          ;****** initialize space for the final segment.  the final segment will less than 32k.
          blobout = notrim(concat(notrim(blobout),notrim(substring(1,finlen,fillstring(32768, " ")))))

DETAIL
          retlen = 1
          offset = 0

          ;*** get the blob segments and concat them into a single variable named good_blob
          ;*** the following while loop is used in case the blob_contents is actually more than 32k
          ;*** in most cases the while loop will only be executed one time because the blob is stored
          ;*** in 32k segments
          while (retlen > 0)
             ;;; ***  this gets a segment of the blob upto 32000 specified by retlen, offset is an accum of retlen
                    retlen = blobget(outbuf, offset, cb.blob_contents)
                    offset = offset + retlen
                    if(retlen!=0)
                       ;*** when dealing with CE_BLOB each row is ended with the tag "ocf_blob"
                       ;*** these tags need to be excluded when the blob segments are re-assembled
          xlen = findstring("ocf_blob",outbuf,1)-1

          if(xlen<1)
            xlen = retlen
          endif

          good_blob = notrim(concat(notrim(good_blob), notrim(substring(1,xlen,outbuf))))

       endif
          endwhile

foot cb.event_id
   newsize = 0
   ;;;***** put the ocf_blob terminator back on the end of the re-assembled blob
   good_blob = concat(notrim(good_blob),"ocf_blob")
          ;;;***** uncompress the re-assembled blob
   blob_un = uar_ocf_uncompress(good_blob, size(good_blob),
                     blobout, size(blobout),newsize )


          ;Output the blof to a file 32000 characters at a time
          offset = 0

          while (offset < size(blobout))
```

```
                ;Output to file...
                print_blob = trim(substring(offset, 32000, blobout))

                if (size(print_blob) > 0)
                        col 0 print_blob
                        row +1
                endif

                offset = offset + 32000

        endwhile

WITH MAXCOL = 32100 , RDBARRAYFETCH = 1, format=undefined

end
go

1_multi_record_blob go
```