ABAP Class to gzip and gunzip

Applies To:

SAP R/3 4.7 and above

Summary

ABAP Implementation of GZIP

By: Otto Frost

Company: Capgemini

Date: 17 March 2006

GZIP / GUNZIP

The zcl_abap_gzip ABAP class implements gzip and gunzip as described in RFC 1952 - GZIP file format specification version 4.3 http://www.faqs.org/rfcs/rfc1952.html.

Gzip/Gunzip is a lossless compressed data format that

- Is independent of CPU type, operating system, file system, and character set, and hence can be used for interchange
- Can compress or decompress a data stream (as opposed to a randomly accessible file) to produce another data stream, using only an a priori bounded amount of intermediate storage, and hence can be used in data communications or similar structures such as Unix filters
- Compresses data with efficiency comparable to the best currently available general-purpose compression methods, and in particular considerably better than the "compress" program
- Can be implemented readily in a manner not covered by patents, and hence can be practiced freely
- Is compatible with the file format produced by the current widely used gzip utility, in that conforming decompressors will be able to read data produced by the existing gzip compressor.

The zcl_abap_gzip ABAP class is intended for use by implementers of software to compress data into gzip format and/or decompress data from gzip format.

If you need more information about gzip, try a google search. On a unix box the command "man gzip" or "gzip --help" may work too. On windows gzip is available in the CygWin package from RedHat. On http://www.gnu.org/ you should be able to find a manual for gzip.

Example code

```
* You may use gui_upload and gui_download to
* transfer the data in binary mode, but that is not included
* in this example. This example first executes ungzip
* and then gzip. Normally you would use gzip or ungzip, not both.

* This example is also available in the method example.

    DATA xstringin TYPE xstring.
    DATA xstringout TYPE xstring.
    DATA file TYPE zcl_abap_gzip=>t_file.
    DATA obj TYPE REF TO zcl_abap_gzip.
    CREATE OBJECT obj.
* file without filename with content AAAAAA
    xstringin = '1F8B0800FBAF0D4402037374040100 7EDE1CAA06000000'.

* a gzipped file with filename testa.txt, with content AAAAAA and a char with hex val x0A
    xstringin = '1F8B0808B43310440203746573746 12E7478740073740 4012E007646C08507000000'.
    xstringout = obj->gunzip( gzip_in = xstringin ) ..
    file = obj->get_file_attributes( ).
    if obj->get_error( ) <> 0.
*      error hadnling code.
    endif.
    xstringin  = xstringout. " content AAAAAA and a char with hex val x0A
    CREATE OBJECT obj.
    obj->set_name( name = file-name ).
    obj->set_compress_level( 9 ).
    xstringout = obj->gzip( raw_in = xstringin ).
    if obj->get_error( ) <> 0.
*      error hadnling code.
    endif.
```

(The cl_abap_gzip handle deflate/inflate/compression, but not member header and trailer in the gzip format. See RFC 1952

ABAP class to gzip and gunzip. The `cl_abap_gzip handle deflate, but not head and tail in gzip format.` This sample was developed to be able to process big files in SAP XI. See the example method for usage.

ABAP Class to gzip and gunzip

Code sample:

```
CLASS zcl_abap_gzip DEFINITION.
* RFC 1952 - GZIP file format specification version 4.3
* http://www.faqs.org/rfcs/rfc1952.html

* Author: Otto Frost

* Possible improvements

* Check input
* ID1, ID2, has correct values
* Set mtime to processingtime
* Handle CRC16
* Improve Error handling

* Tested with
* gnu gzip/gunzip 1.3.5 under cygwin
* winzip 8.1 SR1 can open the produced file
* Test run on SAP XI 3.0 SP14, on windows platform.

* Function modules gui_upload and gui_download may be used for
* up and download of data
* See example method for how to use the class

  PUBLIC SECTION.
    TYPE-POOLS abap.
    TYPES: t_flg(1) TYPE x.
* flags
    CONSTANTS: ftext TYPE i VALUE 0.
    CONSTANTS: fhcrc TYPE i VALUE 1.
    CONSTANTS: fextra TYPE i VALUE 2.
    CONSTANTS: fname TYPE i VALUE 3.
    CONSTANTS: fcomment TYPE i VALUE 4.
*   flags 5, 6, 7 are reserved

* errors
    CONSTANTS: err_invalid_range TYPE i VALUE 1.
    CONSTANTS: err_buffer_overflow   TYPE i  VALUE 2.
    CONSTANTS: err_codepage_converter_init TYPE i VALUE 4.
    CONSTANTS: err_conversion_codepage TYPE i  VALUE 8.
    CONSTANTS: err_parameter_invalid_type TYPE i VALUE 16.
    CONSTANTS: err_parameter_invalid_range TYPE i VALUE 32.
    CONSTANTS: err_xlen TYPE i VALUE 64.
    CONSTANTS: err_name TYPE i VALUE 128.
    CONSTANTS: err_comment TYPE i VALUE 256.
    CONSTANTS: err_gzip TYPE i VALUE 512.
    CONSTANTS: err_crc32 TYPE i VALUE 1024.
    CONSTANTS: err_out_of_bounds TYPE i VALUE 2048.
    CONSTANTS: err_system_exception TYPE i VALUE 4096.
    CONSTANTS: err_cm TYPE i VALUE 8192.

    CONSTANTS: null(1) TYPE x VALUE '00'.
    TYPES:
    BEGIN OF t_file,
           id1(1)  TYPE x,
           id2(1) TYPE x,
           cm(1) TYPE x,
           flg TYPE t_flg,
           mtime(4) TYPE x,
           xfl(1) TYPE x,
           os(1) TYPE x,

           xlen TYPE i,                                 " 2 bytes
           xlenbytes TYPE xstring,

           name TYPE string, " zero terminated
           comment TYPE string, " zero terminated
           namex TYPE xstring, " zero terminated
           commentx TYPE xstring, " zero terminated
           crc16(2) TYPE x,

           compress TYPE xstring,

           crc32(4) TYPE x,
           isize TYPE i,                                " 4 bytes
         END OF t_file .
    CLASS-METHODS example.
    METHODS constructor.
    METHODS: gzip IMPORTING raw_in TYPE xstring RETURNING value(gzip_out) TYPE xstring
      EXCEPTIONS
        zip_parse_error
        cx_parameter_invalid_range
        cx_sy_buffer_overflow.


    METHODS: gunzip IMPORTING gzip_in TYPE xstring RETURNING value(raw_out) TYPE xstring
      EXCEPTIONS
        zip_decompression_error
        cx_parameter_invalid_range
        cx_sy_buffer_overflow.

    METHODS: get_file_attributes RETURNING value(file_attributes) TYPE t_file.
    METHODS: get_error RETURNING value(error) TYPE i.

    METHODS set_name IMPORTING name TYPE string.
    METHODS set_comment IMPORTING comment TYPE string.
    METHODS set_xlenbytes IMPORTING xlenbytes TYPE xstring.
    METHODS set_mtime IMPORTING mtime TYPE xstring.
    METHODS set_compress_level IMPORTING compress_level TYPE i.


* methods not for general use
    METHODS: bit_swap IMPORTING x TYPE t_flg RETURNING value(y) TYPE t_flg.
    METHODS: get_flagbit IMPORTING x TYPE t_flg flag TYPE i RETURNING value(b) TYPE i.
    METHODS: set_flagbit IMPORTING x TYPE t_flg flag TYPE i RETURNING value(y) TYPE i.

  PRIVATE SECTION.
    DATA file TYPE t_file.
    DATA compress_level TYPE i.
    DATA error TYPE i.
    DATA encoding TYPE abap_encoding. " ISO-8859-1

ENDCLASS.                        "zcl_abap_gzip DEFINITION
```

```
*----------------------------------------------------------------------*
*       CLASS zcl_abap_gzip IMPLEMENTATION
*----------------------------------------------------------------------*
*
*----------------------------------------------------------------------*
CLASS zcl_abap_gzip IMPLEMENTATION.
  METHOD constructor.
    DATA sap_codepage TYPE  cpcodepage.

    me->compress_level = 6.

*SCP_CODEPAGE_BY_EXTERNAL_NAME
    CALL FUNCTION 'SCP_CODEPAGE_BY_EXTERNAL_NAME'
      EXPORTING
        external_name       = 'ISO-8859-1'
*   KIND                   = 'H'
      IMPORTING
        sap_codepage        = sap_codepage
    EXCEPTIONS
      not_found            = 1
      OTHERS               = 2
              .
    IF sy-subrc <> 0.
* Config time error, so no nice handling of error
      MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
            WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
    ENDIF.
    me->encoding = sap_codepage.


  ENDMETHOD.                    "constructor
  METHOD get_error.
    error = me->error.
  ENDMETHOD.                    "get_error
  METHOD gzip.
    DATA: x2(2) TYPE x, x4(4) TYPE x.

    DEFINE writex4.    " write xstring
      x4 = &2.
      concatenate &1 x4 into &1 in byte mode.
    END-OF-DEFINITION.

    DEFINE write2.  " write two bytes from integer
      x2 = &2.
      concatenate &1 x2+1(1) x2+0(1) into &1 in byte mode.
    END-OF-DEFINITION.

    DEFINE write4.  " write four bytes from integer
      x4 = &2.
      concatenate &1 x4+3(1) x4+2(1) x4+1(1) x4+0(1) into &1 in byte mode.
    END-OF-DEFINITION.

    DATA oconv TYPE REF TO cl_abap_conv_out_ce.
*     TRY.
    CALL METHOD cl_abap_conv_out_ce=>create
       EXPORTING
*          encoding    = 'UTF-8'
         encoding    = me->encoding
```

```
*         ENDIAN      =
*         REPLACEMENT = '#'
*         IGNORE_CERR = ABAP_FALSE
      RECEIVING
        conv        = oconv.
      .
*      CATCH CX_PARAMETER_INVALID_RANGE .
*      CATCH CX_SY_CODEPAGE_CONVERTER_INIT .
*     ENDTRY.
*

    file-id1 = '1F'.
    file-id2 = '8B'.
    file-cm = '08'.
    file-os = 'FF'. " 255 Unknown ( ABAP )
    file-isize = XSTRLEN( raw_in ).
    TRY.
        CALL METHOD cl_abap_gzip=>compress_binary
          EXPORTING
            raw_in         = raw_in
            raw_in_len     = file-isize
            compress_level = compress_level
           IMPORTING
             gzip_out       = file-compress
*                         GZIP_OUT_LEN   =
            .
      CATCH cx_parameter_invalid_range .
        ADD err_invalid_range TO error.
        ADD err_gzip TO error.
      CATCH cx_sy_buffer_overflow .
        ADD err_buffer_overflow TO error.
        ADD err_gzip TO error.
    ENDTRY.
    file-crc32 = cl_abap_zip=>crc32( raw_in ).

    CONCATENATE file-id1 file-id2
      file-cm file-flg INTO gzip_out IN BYTE MODE.
    write4 gzip_out file-mtime.
    CONCATENATE gzip_out
      file-xfl file-os INTO gzip_out IN BYTE MODE.
    IF 1 = me->get_flagbit( x = file-flg flag = fextra ).
      file-xlen = XSTRLEN( file-xlenbytes ).
      IF file-xlen > 255.
        ADD err_xlen TO error.
      ENDIF.
      write2 gzip_out file-xlen.
      CONCATENATE gzip_out
        file-xlenbytes INTO gzip_out IN BYTE MODE.
    ENDIF.
    IF 1 = me->get_flagbit( x = file-flg flag = fname ).
      TRY.
          CALL METHOD oconv->convert
            EXPORTING
              data   = file-name
*          N      = -1
            IMPORTING
              buffer = file-namex
*          LEN    =
              .
        CATCH cx_sy_codepage_converter_init .
          ADD err_codepage_converter_init TO error.
          ADD err_name TO error.
        CATCH cx_sy_conversion_codepage .
          ADD err_conversion_codepage TO error.
          ADD err_name TO error.
        CATCH cx_parameter_invalid_type .
          ADD err_parameter_invalid_type TO error.
          ADD err_name TO error.
      ENDTRY.
      CONCATENATE gzip_out file-namex null INTO gzip_out IN BYTE MODE.
    ENDIF.
    IF 1 = me->get_flagbit( x = file-flg flag = fcomment ).
      TRY.
          CALL METHOD oconv->convert
            EXPORTING
              data   = file-comment
*           N      = -1
            IMPORTING
              buffer = file-commentx
*           LEN    =
              .
        CATCH cx_sy_codepage_converter_init .
          ADD err_codepage_converter_init TO error.
          ADD err_comment TO error.
        CATCH cx_sy_conversion_codepage .
          ADD err_conversion_codepage TO error.
          ADD err_comment TO error.
        CATCH cx_parameter_invalid_type .
          ADD err_parameter_invalid_type TO error.
          ADD err_comment TO error.
      ENDTRY.
      CONCATENATE gzip_out file-commentx null INTO gzip_out IN BYTE MODE.
    ENDIF.
    CONCATENATE gzip_out file-compress INTO gzip_out IN BYTE MODE.
    CLEAR file-compress.
    write4 gzip_out file-crc32.
    write4 gzip_out file-isize.
  ENDMETHOD.                    "gzip

  METHOD gunzip.
    TRY.
        DATA: offset  TYPE i.

        DEFINE next.   " move offset
          offset = offset + &1.
        END-OF-DEFINITION.

        DATA: l1(1) TYPE x, h1(1) TYPE x, l2(1) TYPE x, h2(1) TYPE x, xstr TYPE xstring.
        DEFINE read2.  " read two bytes as integer and move offset
          l1 = gzip_in+offset(1). offset = offset + 1.
          h1 = gzip_in+offset(1). offset = offset + 1.
          concatenate h1 l1 into xstr in byte mode.
          &1 = xstr.
        END-OF-DEFINITION.

        DEFINE read4.  " read four bytes as integer and move offset
```

```
          l1 = gzip_in+offset(1). offset = offset + 1.
          h1 = gzip_in+offset(1). offset = offset + 1.
          l2 = gzip_in+offset(1). offset = offset + 1.
          h2 = gzip_in+offset(1). offset = offset + 1.
          concatenate h2 l2 h1 l1 into xstr in byte mode.
          &1 = xstr.
        END-OF-DEFINITION.

* We convert all names from xstring into string
        DATA: conv TYPE REF TO cl_abap_conv_in_ce.
        TRY.
            conv = cl_abap_conv_in_ce=>create( encoding = encoding ).
          CATCH cx_parameter_invalid_range.
            ADD err_parameter_invalid_range TO error.
          CATCH cx_sy_codepage_converter_init.
            ADD err_codepage_converter_init TO error.
        ENDTRY.

        DATA: msdos_date TYPE i, msdos_time TYPE i.
*  FIELD-SYMBOLS:  <file> TYPE t_file,
*                  <ext>  TYPE t_ext.
        DATA b TYPE i. " BIT

        DATA: max_length TYPE i.
        DATA file TYPE t_file.
        DATA crc32 TYPE i.

        file-id1 = gzip_in+offset(1). next 1.
        file-id2 = gzip_in+offset(1). next 1.
        file-cm = gzip_in+offset(1). next 1.
        IF file-cm <> '08'.
          ADD err_cm TO error.
        ENDIF.
        file-flg = gzip_in+offset(1). next 1.
        read4 file-mtime.
        file-xfl = gzip_in+offset(1). next 1.
        file-os = gzip_in+offset(1). next 1.
*     DO 8 TIMES.
*       GET BIT sy-index OF file-flg INTO b.
*       WRITE b NO-GAP.
*     ENDDO.
        b = me->get_flagbit( x = file-flg flag = fextra ).
        IF b = 1.
          read2 file-xlen.
          file-xlenbytes = gzip_in+offset(file-xlen). next file-xlen.
        ENDIF.
        b = me->get_flagbit( x = file-flg flag = fname ).
        IF b = 1.
          WHILE gzip_in+offset(1) <> '00'.
            CONCATENATE file-namex gzip_in+offset(1) INTO file-namex IN BYTE MODE.
            next 1.
          ENDWHILE.
          next 1.                                         " skip 00
          TRY.
              conv->convert( EXPORTING input = file-namex IMPORTING data = file-name ).
            CATCH cx_sy_conversion_codepage.
              ADD err_conversion_codepage TO error.
              ADD err_name TO error.
            CATCH cx_sy_codepage_converter_init.
              ADD err_codepage_converter_init TO error.
              ADD err_name TO error.
            CATCH cx_parameter_invalid_type.
              ADD err_parameter_invalid_type  TO error.
              ADD err_name TO error.
          ENDTRY.
        ENDIF.
        b = me->get_flagbit( x = file-flg flag = fcomment ).
        IF b = 1.
          WHILE gzip_in+offset(1) <> '00'.
            CONCATENATE file-commentx gzip_in+offset(1) INTO file-commentx IN BYTE MODE.
            next 1.
          ENDWHILE.
          next 1.                                          " skip 00
          TRY.
              conv->convert( EXPORTING input = file-commentx IMPORTING data = file-
comment ).
            CATCH cx_sy_conversion_codepage.
              ADD err_conversion_codepage TO error.
              ADD err_comment TO error.
            CATCH cx_sy_codepage_converter_init.
              ADD err_codepage_converter_init TO error.
              ADD err_comment TO error.
            CATCH cx_parameter_invalid_type.
              ADD err_parameter_invalid_type  TO error.
              ADD err_comment TO error.
          ENDTRY.
        ENDIF.
        b = me->get_flagbit( x = file-flg flag = fhcrc ).
        IF b = 1.
          read2 file-crc16.
        ENDIF.
        max_length = XSTRLEN( gzip_in ).
        max_length = max_length - offset.
        max_length = max_length  - 8. " trailer length
        file-compress = gzip_in+offset(max_length). next max_length.
        read4 file-crc32.
        read4 file-isize.
        TRY.
            cl_abap_gzip=>decompress_binary(
              EXPORTING gzip_in     = file-compress
                        gzip_in_len = max_length
              IMPORTING raw_out     = raw_out ).
          CATCH cx_parameter_invalid_range.
            ADD err_parameter_invalid_range TO error.
            ADD err_gzip TO error.
          CATCH cx_sy_buffer_overflow.
            ADD err_buffer_overflow TO error.
            ADD err_gzip TO error.
        ENDTRY.
        crc32 = cl_abap_zip=>crc32( raw_out ).
        IF crc32 <> file-crc32.
*      RAISE zip_decompression_error.
          ADD err_crc32 TO error.
        ENDIF.
        CLEAR file-compress.
        me->file = file.
      CATCH cx_sy_range_out_of_bounds.
        ADD err_out_of_bounds TO error.
    ENDTRY.
  ENDMETHOD.                              "gunzip
  METHOD bit_swap.
* in abap bytes are indexed
* 1,2,3,4,5,6,7,8
* in gzip spec
* 7,6,5,4,3,2,1,0
* why order is reversed
* and 1 added to index in get_flagbit method
    DATA b TYPE i. " BIT
    DATA i TYPE i.
    i = 8.
    DO 8 TIMES.
      GET BIT sy-index OF x INTO b.
      SET BIT i OF y TO b.
      i = i - 1.
    ENDDO.
  ENDMETHOD.                              "bit_swap
  METHOD get_flagbit.
    DATA byte(1) TYPE x.
    DATA index TYPE i.
    byte = me->bit_swap( x = x ).
    index = 1 + flag.
    GET BIT index OF byte INTO b.
  ENDMETHOD.                              "get_flagbit
  METHOD set_flagbit.
    DATA index TYPE i.
    DATA byte(1) TYPE x.
    index = flag + 1.
    byte = me->bit_swap( x = x ).
    SET BIT index OF byte.
    byte = me->bit_swap( x = byte ).
    y = byte.
  ENDMETHOD.                              "set_flagbit
  METHOD get_file_attributes.
    file_attributes = me->file.
  ENDMETHOD.                              "get_file_attributes

  METHOD set_name.
*     methods set_name importing name type string.
    file-name = name.
    file-flg = me->set_flagbit( x = file-flg flag = fname ).
  ENDMETHOD.                              "set_name
  METHOD set_comment.
*     methods set_comment importing comment type string.
    file-comment = comment.
    file-flg = me->set_flagbit( x = file-flg flag = fcomment ).
  ENDMETHOD.                              "set_comment
  METHOD set_xlenbytes.
*     methods set_xlenbytes importing xlenbytes type xstring.
    file-xlenbytes = xlenbytes.
    file-flg = me->set_flagbit( x = file-flg flag = fextra ).
  ENDMETHOD.                              "set_xlenbytes
  METHOD set_mtime.
    file-mtime = mtime.
  ENDMETHOD.                              "set_mtime
  METHOD set_compress_level.
    me->compress_level = compress_level.
    IF compress_level = 9.
      file-xfl = '02'.
    ELSEIF compress_level = 1.
      file-xfl = '04'.
    ENDIF.
  ENDMETHOD.                    "set_compress_level
  METHOD example.
* You may use gui_upload and gui_download to
* transfer the data, but that is not included
* in this example
    DATA xstringin TYPE xstring.
    DATA xstringout TYPE xstring.
    DATA file TYPE zcl_abap_gzip=>t_file.
    DATA obj TYPE REF TO zcl_abap_gzip.
    CREATE OBJECT obj.
* file without filename with content AAAAAA
    xstringin = '1F8B0800FBAF0D4402037374040100 7EDE1CAA06000000'.

* a gzipped file with filename testa.txt, with content AAAAAA and a char with hex val x0A
    xstringin = '1F8B0808B433104402037465737461 2E7478740073740401 2E007646C08507000000'.
    xstringout = obj->gunzip( gzip_in = xstringin ) .
    file = obj->get_file_attributes( ).
    if obj->get_error( ) <> 0.
*     error hadnling code.
    endif.
    xstringin  = xstringout.
    CREATE OBJECT obj.
    obj->set_name( name = file-name ).
    obj->set_compress_level( 9 ).
    xstringout = obj->gzip( raw_in = xstringin ).
    if obj->get_error( ) <> 0.
*     error hadnling code.
    endif.

  ENDMETHOD.                    "example

ENDCLASS.                    "zcl_abap_gzip IMPLEMENTATION
```

Author Bio

A developer working in various languages and systems. Started with ABAP in 2002. Senior consultant in EDI integrations. He is also a certified SAP XI consultant.

Disclaimer & Liability Notice